© 2014 Saiprasad Ravishankar

ADAPTIVE SPARSE REPRESENTATIONS AND THEIR APPLICATIONS

BY

SAIPRASAD RAVISHANKAR

DISSERTATION

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Electrical and Computer Engineering
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2014

Urbana, Illinois

Doctoral Committee:

Professor Yoram Bresler, Chair
Professor Minh N. Do
Associate Professor Brad Sutton
Associate Professor Olgica Milenkovic
Professor Jeffrey A. Fessler, University of Michigan

ABSTRACT

The sparsity of signals and images in a certain transform domain or dictionary has been exploited in many applications in signal processing, image processing, and medical imaging. Analytical sparsifying transforms such as Wavelets and DCT have been widely used in compression standards. Recently, the data-driven learning of synthesis sparsifying dictionaries has become popular especially in applications such as denoising, inpainting, and compressed sensing. While there has been extensive research on learning synthesis dictionaries and some recent work on learning analysis dictionaries, the idea of learning sparsifying transforms has received no attention. In the first part of this thesis, we study the sparsifying transform model and its relationship to prior linear sparse models. Then, we propose novel problem formulations for learning square sparsifying transforms from data. The proposed algorithms for transform learning alternate between a sparse coding step and a transform update step, and are highly efficient. Specifically, as opposed to sparse coding in the synthesis or noisy analysis models which is NP-hard, the sparse coding step in transform learning can be performed exactly and cheaply by zeroing out all but a certain number of nonzero transform coefficients of largest magnitude. The transform update step is performed using iterative conjugate gradients. The proposed algorithms give rise to well-conditioned square sparsifying transforms in practice. We show the superiority of our approach over analytical sparsifying transforms such as the DCT for signal and image representation. We also show promising performance in signal denoising using the learned sparsifying transforms. The proposed approach is much faster than previous approaches involving learned synthesis, or analysis dictionaries.

Next, we explore a specific structure for learned sparsifying transforms, that enables efficient implementations. Following up on the idea of learning square sparsifying transforms, we propose novel problem formulations for learning doubly sparse transforms for signals or image patches. These transforms are a product of a fixed, fast analytic transform such as the DCT, and an adaptive matrix constrained to be sparse. Such transforms can be learned, stored, and implemented efficiently. We show the superior promise of our learned doubly sparse transforms as compared to analytical sparsifying transforms such as the DCT or Wavelets for image representation. Adapted doubly sparse transforms also generalize better than the unstructured (or non-sparse) transform. We show promising performance and speedups in image denoising using the learned doubly sparse transforms compared to approaches involving learned synthesis dictionaries such as the K-SVD algorithm.

In the third part of this thesis, we further develop the alternating algorithms for learning unstructured (non-sparse) well-conditioned, or orthonormal square sparsifying transforms. While, in the first part of the thesis, we provided an iterative method involving conjugate gradients for the transform update step, in this part, we instead derive efficient and analytical closed-form solutions for transform update. Importantly, we establish that the proposed algorithms are globally convergent to the set of local minimizers of the non-convex transform learning problems. In practice, our algorithms are shown to be insensitive to initialization.

In the next part of the thesis, we focus on compressed sensing (CS), which exploits the sparsity of images or image patches in a transform domain or synthesis dictionary to reconstruct images from highly undersampled or compressive measurements. Specifically, we focus on the subject of blind compressed sensing, where the underlying sparsifying transform is unknown a priori, and propose a framework to simultaneously reconstruct the underlying image(s)/volume(s) as well as the square sparsifying transform from highly undersampled measurements. The proposed block coordinate descent type algorithms involve highly efficient closed-form optimal updates. Importantly, we prove that although the proposed blind compressed sensing formulations are highly nonconvex, our algorithms converge to the set of critical points of the objectives defining the formulations. We illustrate the usefulness of the proposed framework for magnetic resonance image (MRI) reconstruction from highly undersampled k-space measurements. As compared to previous state-of-the-art methods involving the synthesis model, our approach is 10x faster for reconstructing 2D MR images, while also providing promising reconstruction quality. The proposed transform-based blind compressed sensing has the potential to revolutionize medical imaging technologies by highly accelerating both the imaging and image reconstruction processes.

In the fifth part of this thesis, we study the design of sampling schemes for compressed sensing MRI. The (pseudo) random sampling schemes used most often for CS may have good theoretical asymptotic properties; however, with limited data they may be far from optimal. Therefore, we propose a novel framework for improved adaptive sampling schemes for highly undersampled CS MRI. While the proposed framework is general, we apply it with some recent MRI reconstruction algorithms. Numerical experiments demonstrate that our adaptive sampling scheme can provide significant improvements in image reconstruction quality for MRI compared to non-adapted methods.

In the next part of the thesis, we develop a methodology for online learning of square sparsifying transforms. Such online learning is particularly useful when dealing with big data, and for signal processing applications such as real-time sparse representation and denoising. The proposed transform learning algorithms are shown to have a much lower computational cost than online synthesis dictionary learning. In practice, the sequential learning of a sparsifying transform typically converges much faster than batch mode transform learning. Preliminary experiments show the usefulness of the proposed schemes for sparse representation (compression), and denoising. We also prove that although the associated optimization problems are non-convex, our online transform learning algorithms are guaranteed to converge to the set of stationary points of the learning problem. The guarantee relies on few (easy to verify) assumptions.

In the seventh part of this thesis, we propose a novel convex formulation for doubly sparse square transform learning. The proposed formulation has similarities to traditional least squares optimization with ℓ_1 regularization. Our convex learning algorithm is a modification of FISTA, and is guaranteed to converge to a global optimum, and moreover converges quickly. We also study two non-convex variants of the proposed convex formulation, and provide local convergence proof for the algorithm for one of them. These proposed non-convex variants use the ℓ_0 "norm" for measuring the sparsity of the transform and/or sparse code. We show the superior promise of our learned transforms here as compared to analytical sparsifying transforms such as the DCT for image representation. In these examples, the performance is some-times comparable to the previously proposed non-convex (non guaranteed) doubly sparse transform learning schemes.

While we studied the learning of square transforms in the initial parts of the thesis, in the eighth part of the thesis, we instead briefly study the learning of tall or overcomplete sparsifying transforms from data. We propose various penalties that control the sparsifying ability, condition number, and incoherence of the learned transforms. Our alternating algorithm for overcomplete transform learning converges empirically, and significantly improves the quality of the learned transform over the iterations. We present examples demonstrating the promising performance of adaptive overcomplete transforms over adaptive overcomplete synthesis dictionaries learned using the popular K-SVD algorithm, in the application of image denoising. The overcomplete transforms also denoise better than adaptive square transforms.

In the final part of the thesis, we explore the idea of learning efficient structured overcomplete sparsifying transforms. Since natural images typically contain diverse textures that cannot be sparsified well by a single transform, we therefore propose a union of sparsifying transforms model. Sparse coding in this model reduces to a form of transform-domain clustering. This makes the model appealing for classification tasks. The proposed model is also equivalent to a structured overcomplete sparsifying transform model with block cosparsity, dubbed OCTOBOS. The alternating algorithm introduced for learning such transforms involves simple closed-form solutions. A theoretical analysis provides a convergence guarantee for this algorithm. It is shown to be globally convergent to the set of partial minimizers of the non-convex OCTOBOS (or, union of transforms) learning problem. We also show that under certain conditions, the algorithm converges to the set of stationary points of the overall objective. When applied to images, the algorithm learns a collection of well-conditioned square transforms, and a good clustering of patches or textures. The resulting sparse representations for the images are much better than those obtained with a single learned transform, or with analytical transforms. We show the promising performance of the proposed approach in image denoising, which compares quite favorably with approaches involving a single learned square transform or an overcomplete synthesis dictionary, or Gaussian mixture models. The proposed denoising method is also faster than the synthesis dictionary based approach.

To the Universe.

ACKNOWLEDGMENTS

I would like to thank my wonderful adviser, Prof. Yoram Bresler, for his highly insightful guidance and the many long, engaging, and rewarding discussions, comments, and suggestions throughout this work. Prof. Bresler has always been extremely generous towards me with his time and support. I have learned so many things from him, both academically and otherwise. I would also like to thank the other committee members, namely Professors Olgica Milenkovic, Brad Sutton, Minh Do, and Jeffrey Fessler, for their feedback on my work. They have also been an inspiration to me as teachers or as researchers.

I wish to thank secretary Ms. Peggy Wells for being very helpful and kind on several occasions during my study here. I particularly wish to thank her for always welcoming me into her office with a smile. I thank my colleagues Bihan Wen, Kiryung Lee, and Luke Pfister for their various collaborations with me, and for the many useful discussions or brainstorming sessions that we had during the course of my PhD. I would also like to thank other colleagues in Prof. Bresler's group, such as Yanjun Li and Elad Yarkony, for being awesome group mates. Being a part of this group has been a lot of fun, and I have learned many things from my colleagues.

I also owe a lot of thanks to my parents and brother for their support, love, and guidance.

Last but not least, I would like to acknowledge the support for this work from the National Science Foundation (NSF) under grants CCF 06-35234, CCF-1018660, and CCF-1320953.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Sparse representation of signals has become widely popular in recent years. It is well known that natural signals and images have an essentially sparse representation (few significant non-zero coefficients) in analytical transform domains such as Wavelets [12] and discrete cosine transform (DCT). This property has been exploited in designing various compression algorithms and in compression standards such as JPEG2000 [13].

While transforms are a classical tool in signal processing, alternative models have also been studied for sparse representation of data. Two such well-known models are the synthesis and analysis models [14]. More recently, attention has turned to adapting these models to data. These approaches are known in the literature as synthesis dictionary learning [15, 16, 17, 18, 19, 20] and analysis dictionary learning [21, 22, 23, 24, 25, 26]. Surprisingly, the adaptation or learning of transforms has received no attention. In this thesis, we develop formulations aimed at learning sparsifying transforms from data. In the following, we discuss the synthesis and analysis models and their learning, and elucidate their drawbacks. This discussion will also serve to highlight the fundamental differences between the transform model and the synthesis/analysis models. The former will be shown to possess important advantages. These advantages will be repeatedly exploited in the various methods in this thesis.

1.1 Synthesis Model and Its Adaptation to Data

A popular sparse representation model is the *synthesis model*, which states that a signal $y \in \mathbb{R}^n$ may be represented as a linear combination of a small number of atoms/columns from a dictionary $D \in \mathbb{R}^{n \times K}$ [14, 27]. This means $y = Dx$, where $x \in \mathbb{R}^K$ is sparse with $\|x\|_0 \ll K$, and the l_0 "norm" counts the number of non-zeros in x. In practice, "real world" signals are expected to deviate from this model. Therefore, the data is more generally assumed to satisfy $y = Dx + \xi$, where ξ is an error or noise term in the signal domain [27]. When $n = K$, and D is full rank, the dictionary D is said to be a basis, whereas when $K > n$, the dictionary is said to be overcomplete.

The approach of representing a signal in a synthesis dictionary D has received considerable attention recently. Specifically, the problem shown in (1.1), of extracting the sparse representation of a signal y in a synthesis dictionary D, has been studied in great detail in recent years [28, 29, 30, 31], and is popularly known as the (synthesis) sparse coding problem [1].

$$\min_{x} \|y - Dx\|_2^2 \;\; s.t. \;\; \|x\|_0 \leq s \tag{1.1}$$

Here, parameter s represents the desired sparsity level. This synthesis sparse coding problem is known to be NP-hard (Non-deterministic Polynomial-time hard) [32, 33] and therefore computationally infeasible in general. In practice, the problem can be solved approximately by greedy [34, 35, 36], or relaxation algorithms [37, 38, 39]. Under certain conditions, some of these algorithms can be guaranteed to provide the correct solution, or provide it with high probability [31, 28, 39, 29, 30, 40, 36]. However, in applications, especially those involving learning the models from data, these conditions are often violated. Other algorithms, for which similar theoretical performance guarantees may not be available, often provide even better empirical performance [38, 41, 42, 43, 44]. All these various algorithms are, however, computationally expensive in practice, particularly for large-scale problems.

The process of obtaining a sparse representation for a signal/image requires explicit knowledge of the synthesis dictionary D. Many analytical dictionaries have been developed for sparse image representation such as the Ridgelet [45], Contourlet [46], and Curvelet [47] dictionaries.

The idea of learning a synthesis dictionary from training signals has also been exploited [15, 16, 17, 18]. Given a matrix $Y \in \mathbb{R}^{n \times N}$ whose columns represent training signals, the problem of learning a dictionary D that gives

[1]Alternative formulations swap the constraint and cost, or use a Lagrangian formulation.

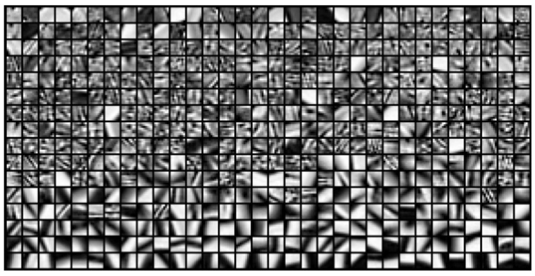


Figure 1.1: A synthesis dictionary learned from MR images. The columns of the dictionary are shown as patches.

a sparse representation for the signals in Y, can be formulated as follows [17].

$$\text{(P0s)} \quad \min_{D,X} \|Y - DX\|_F^2 \;\; s.t. \;\; \|X_i\|_0 \le s \;\; \forall \; i \tag{1.2}$$

Here, the columns X_i of matrix $X \in \mathbb{R}^{K\times N}$ with maximum allowed sparsity level s are the sparse representations, or sparse codes, of the signals/columns in Y, and $\|\cdot\|_F$ denotes the Frobenius norm. The columns of the learned dictionary in (P0s) are typically constrained to be of unit norm (or normalized) in order to avoid the scaling ambiguity [48]. While Problem (P0s) uses the l_0 "norm" for sparsity, one could also alternatively use its convex relaxation, the l_1 norm [20].

Problem (P0s) is to minimize the fitting error of the training signals with respect to the dictionary D subject to sparsity constraints. This problem too is NP-hard (for each fixed D, it requires the solution of N sparse coding problems), and there are no known polynomial-time algorithms for its exact solution under practically relevant conditions. Nonetheless, several popular heuristics have been developed for the solution of (P0s), or its variations, in recent years [16, 17, 18, 19, 20, 49, 50, 51]. The algorithms typically alternate between finding the dictionary D (dictionary update step) [2], and the sparse representations X (sparse coding step). The sparse coding step is always solved with fixed D. Of the various algorithms, the K-SVD algorithm [17] has been especially popular in applications. Figure 1.1 [52] shows a synthesis dictionary learned from magnetic resonance images using the K-SVD algorithm, with the columns of the dictionary shown as patches.

[2]Some algorithms (e.g., K-SVD) also update the non-zero coefficients of the sparse code X in the dictionary update step.

Unfortunately, since Problem (P0s) is non-convex, methods such as K-SVD [3] can get easily caught in local minima or even saddle points [53]. Moreover, the sparse coding step in such methods is computationally expensive. Recent theoretical analysis of dictionary identification using an l_1 relaxation of the sparsity penalty on X provides conditions under which the desired solution is a local minimum of the cost function [48] [54], and under restrictive assumptions (e.g., square, non-singular dictionary, Bernoulli-Gaussian X, and noiseless data) an algorithm recovering the correct dictionary with high probability has been proposed [55]. However, no proof exists to date for the convergence to global or local minimum of K-SVD or other popular synthesis dictionary learning algorithms.

Adaptive synthesis dictionaries have been shown to be useful in a variety of applications such as image/video denoising, image/video inpainting, deblurring, demosaicing [1, 56, 57, 58, 59, 60, 61], clustering and classification [62], and tomography [63]. Recently, we have shown impressive performance for learned dictionaries in magnetic resonance imaging (MRI) involving highly undersampled k-space data [5, 64].

1.2 Analysis Model and Its Adaptation to Data

Another model for sparse representation of data is the *analysis model* [14]. This model suggests that given a signal $y \in \mathbb{R}^n$ and operator $\Omega \in \mathbb{R}^{m \times n}$, the representation $\Omega y \in \mathbb{R}^m$ is sparse, i.e., $\|\Omega y\|_0 \ll m$ [26, 23]. Here, Ω is known as the analysis dictionary, since it "analyzes" the signal y to produce a sparse result. The zeros of Ωy essentially define the subspace to which the signal belongs, and the total number of these zeros is called co-sparsity [26, 23]. A well-known analysis dictionary is the finite difference dictionary. Elad et al. [14] derived conditions for the equivalence of analysis and synthesis based priors.

When the given signal $y \in \mathbb{R}^n$ is contaminated with noise, the analysis model is extended as $y = q + \xi$, with Ωq being sparse, and ξ representing the noise in the signal domain that is assumed to be small [26, 23]. We refer to this as the *noisy signal analysis model.* Given the noisy signal y

[3]In fact, the K-SVD method, although popular, does not have any convergence guarantees.

and dictionary Ω, the problem of recovering the clean signal q is as follows [26, 23].

$$\min_{q} \|y - q\|_2^2 \;\; s.t. \;\; \|\Omega q\|_0 \leq m - t \tag{1.3}$$

Here, parameter t represents the minimum allowed co-sparsity. This problem is called the analysis sparse coding problem [26]. It can also be viewed as a denoising scheme. This problem too is NP-hard, just like sparse coding in the synthesis case. Similarly to sparse coding in the synthesis model, approximate algorithms exist for analysis sparse coding [65, 23, 66, 67, 26, 68, 25, 69], which however, tend to be computationally expensive. Note that when the analysis dictionary Ω is square and non-singular, we have $q = \Omega^{-1}v$ for some sparse v, and the problem of finding q above is identical to a synthesis sparse coding problem of finding v, where $D = \Omega^{-1}$ is the synthesis dictionary. While (1.3) uses the l_0 "norm" for sparsity, some authors [25] alternatively consider a convex relaxation of the problem in which they replace it with an l_1 norm that they add as a penalty in the cost.

While there has been considerable research on the learning of synthesis dictionaries, the idea of learning analysis dictionaries has received only recent attention. Peyré et al. [21] learn a dictionary in a sparsity-promoting analysis-type prior. Analysis dictionary learning has been pursued in some recent papers. Given the training matrix $Y \in \mathbb{R}^{n \times N}$, Ophir et al. [24] and Yaghoobi et al. [22] minimize the l_0 and l_1 norm of ΩY respectively. The goal in these papers is to learn an analysis dictionary that *directly sparsifies* the training data Y. However, the convergence behavior of these learning algorithms has not been analyzed, even in the case when l_1 relaxation is used for sparsity.

Some authors alternatively consider analysis dictionary learning with the noisy signal analysis model [23, 25, 26]. Rubinstein et al. [23, 26] proposed the following formulation for the noisy signal case that involves the l_0 "norm" for sparsity.

$$\text{(P0a)} \;\; \min_{\Omega, Q} \|Y - Q\|_F^2 \;\; s.t. \;\; \|\Omega Q_i\|_0 \leq m - t \;\; \forall \; i \tag{1.4}$$

The columns of matrix ΩQ are constrained to have a co-sparsity level of at least t. In (P0a), the rows of Ω are also typically constrained to be of unit norm [26, 25]. Yaghoobi et al. [25, 70] relax the l_0 "norm" in (P0a) to an

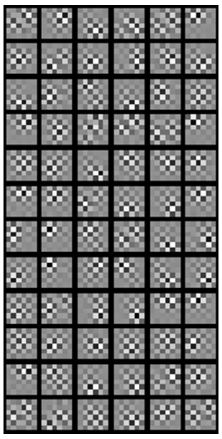


Figure 1.2: An analysis dictionary learned from the 'house' image.

l_1 norm and add it as a penalty in the cost. They also additionally restrict Ω to be a tight frame ($\Omega^T\Omega = I$, where $(.)^T$ denotes the matrix transpose operation, and I is the $n \times n$ identity matrix).

Problem (P0a) is non-convex and NP-hard. However, heuristics have been proposed for its solution such as the analysis K-SVD [26, 23] algorithm, that alternates between updating Ω and Q (analysis sparse coding), and employs a backward-greedy method for updating Q.

Other analysis dictionary learning approaches have also been proposed very recently [71, 72]. Unfortunately, no convergence guarantees exist for the various analysis dictionary learning algorithms. Indeed, because these algorithms (e.g., analysis K-SVD) tend to be similar to those proposed in the synthesis case, we expect that they can get similarly stuck in bad local minima. Importantly, these algorithms tend to be computationally expensive in practice.

Figure 1.2 [4] shows an analysis dictionary learned from the well-known 'house' image [59] using the analysis K-SVD algorithm, with the rows of the dictionary shown as patches.

The promise of learned analysis dictionaries in signal and image processing applications or in medical imaging has not been fully explored or analyzed. Yaghoobi et al. [25, 70], for example, showed that their learned operator denoises not much better than even a fixed analytical dictionary.

[4]This image is taken from Elad's presentation at the SPARS'11 Workshop [73, 23].

1.3 Transform Model for Sparse Representation

In this thesis, we pursue a generalized analysis model for sparse representation of data, which we call the transform model. This model suggests that a signal $y \in \mathbb{R}^n$ is *approximately sparsifiable* using a transform $W \in \mathbb{R}^{m\times n}$, that is $Wy = x + e$ where $x \in \mathbb{R}^m$ is sparse, i.e., $\|x\|_0 \ll m$, and e is the representation error or residual in the *transform domain* that is assumed to be small. Note that this is in contrast with the synthesis and noisy signal analysis models, which allow for errors only in the signal domain. Well-known examples of sparsifying transforms for natural signals and images include Wavelets [12], discrete cosine transform (DCT), Ridgelets [45], Contourlets [46], Curvelets [47], etc. When $m = n$, the $W \in \mathbb{R}^{n\times n}$ is called a square transform. The case when $m > n$ refers to a "Tall" or overcomplete transform.

The transform model suggests that signals can be approximately sparse in the transform domain. This can be viewed as a generalization of the analysis model with Ωy exactly sparse. Additionally, unlike the analysis model in which the sparse representation Ωy lies in the range space of Ω, the sparse representation x in the transform model is not constrained to lie in the range space of W. The generalization allows the transform model to include a wider class of signals within its ambit than the analysis model. The transform model $Wy \approx x$ with x sparse, is also more general than the notion of compressibility [74] applied to Wy, where the assumption would be that the coefficients of Wy, when arranged in non-increasing magnitude order, follow a power law decay. The transform model only assumes that the residual $Wy - x$ has small energy compared to the energy in x.

The reason we have chosen the name "transform model" for our approach is because the assumption $Wy \approx x$ has been traditionally used in transform coding (with orthonormal transforms), and the concept of transform coding is older [75] and pre-dates the terms analysis and synthesis [76].

The transform model is less restrictive (or, more general) not only than the analysis model, but also than the noisy signal analysis model (in the overcomplete case) in the following sense. If a signal y satisfies a noisy signal analysis model with analysis dictionary W and sparse code z' ($= Wq$ for some q), then it also satisfies a corresponding transform model with transform W and sparse code z'. However, the converse is not true in general. That is, a particular transform model for signal y (characterized here by a pair (W, x), with x the sparse code) need not translate to a corresponding noisy signal analysis model. To demonstrate this point, we use for convenience, the same symbol W for both the transform and analysis operator.

First, if we assume that the signal y satisfies the noisy signal analysis model, we have the representation $y = q + \xi$ with $z' = Wq$ being sparse. This implies that

$$Wy = W(q + \xi) = Wq + W\xi = z' + e' \tag{1.5}$$

where $e' = W\xi$ is the error in the transform domain of W. Thus, z' is a transform code with error e' for the noisy y. Hence, it is true that if a signal y satisfies the noisy signal analysis model, it also satisfies a corresponding transform model. However, the converse is not true, in general. That is, for a given signal y, if the transform model $Wy = x + e$ holds with sparse x, then the relation $y = q' + \xi'$, with $Wq' = x$ need not hold for any error ξ'. This is because the transform model does not restrict its sparse code x to lie in the range space of W (i.e., $Wq' = x$ need not be true for any q', especially for a tall/overcomplete transform W). Hence, if a signal satisfies the (overcomplete) transform model, it need not satisfy a corresponding noisy signal analysis model. In this particular sense, the noisy signal analysis model is restrictive compared to the transform model.

In Chapter 2, we will also consider an explicit extension of the transform model for the "noisy signal" case, that makes it even more general.

We note that in the aforementioned arguments (after (1.5)), the relation $Wq' = x$ can be satisfied when the transform W is square and full rank (invertible), in which case $q' = W^{-1}x$. However, although for the square and invertible case, the noisy signal analysis model translates to a corresponding transform model and vice-versa, the error ξ in the signal domain in the noisy signal analysis model becomes $e' = W\xi$ in the transform domain and conversely, the error e in the transform domain for the transform model becomes $\xi' = W^{-1}e$ in the signal domain. Thus, even small errors in one model can be amplified, when translated to the other model for a poorly conditioned matrix W. (That is, $\|W\xi\|_2$ or its "normalized" version $\|W\xi\|_2 / \|Wy\|_2$, and $\|W^{-1}e\|_2$ or its "normalized" version $\|W^{-1}e\|_2 / \|y\|_2$, can be large.) This indicates that the noisy signal analysis model can translate to a bad transform model and vice-versa.

When a sparsifying transform W is known for the signal y, the process of obtaining a sparse code x of given sparsity s involves solving the following problem, which we call the transform sparse coding problem [5].

$$\min_{x} \|Wy - x\|_2^2 \;\; s.t. \;\; \|x\|_0 \leq s \tag{1.6}$$

This problem involves the minimization of the transform domain residual. The solution $\hat{x}$ is obtained exactly by thresholding the product Wy and retaining the s components of largest magnitude, setting the remaining components to zero. In contrast, sparse coding with the synthesis or analysis dictionaries involves in general an NP-hard problem. Given the transform W and sparse code x, we can also recover a least squares estimate (we consider the least squares estimate here because of its simplicity) of the true signal y by minimizing $\|Wy - x\|_2^2$ over all $y \in \mathbb{R}^n$. The recovered signal is then simply $W^{\dagger}x$, where $W^{\dagger}$ is the pseudo-inverse of W. Thus, unlike the previous models, the transform model allows for exact and fast computations, a property that has been exploited heavily in the context of analytical sparsifying transforms.

1.4 Brief Summary of Thesis Contributions

While analytical sparsifying transforms such as the Wavelets, DCT, and the discrete Fourier transform (DFT) have been extensively used in many applications such as compression [13], denoising, and compressed sensing [3, 78], adapting the sparsifying transform to the data could improve its performance significantly in these and other applications. In this adaptive setting, given a matrix Y of training signals, we would like to learn a transform W such that the *sparsification error* $\|WY - X\|_F^2$ (i.e., the modeling error in the transform model) is minimized. Here, X is again a matrix (unknown) containing the sparse codes of the training signals as columns. This idea of learning sparsifying transforms is the main subject of this thesis. We will consider various schemes for learning square as well as overcomplete sparsi-

[5] The transform sparse coding solution can also be alternatively written as $E(Wy)$, where $E(\cdot)$ is the Moreau-Yosida regularization [77] of the indicator function of the ℓ_0 ball $\{x \in \mathbb{R}^m : \|x\|_0 \leq s\}$. This interpretation is not surprising given that the transform model generalizes the analysis model.

fying transforms. We will refer to some of the sparsifying transforms in this work as *structured* transforms, while others will be referred to as *unstructured* transforms (either square, or overcomplete). The main difference is that the former are typically much more constrained than the latter. The imposition of more constraints on the learned transform leads to certain advantages (e.g., computational or other advantages, over unstructured transforms) in some scenarios. The usefulness of the various proposed transform learning schemes will be demonstrated in applications such as sparse image representation (compression), classification, denoising, and compressed sensing. We will also consider 'big data' scenarios and online learning in our applications. As opposed to prior work involving synthesis or analysis dictionary learning, we provide convergence guarantees for many of the methods in this work. Our schemes are also typically much faster than those involving learned synthesis, or learned analysis dictionaries. Apart from the subject of sparsifying transform learning, we also briefly study the design of adaptive sampling schemes for compressed sensing MRI (i.e., with sampling rates below the Nyquist rate) in this work. The proposed sampling framework can in fact be adapted to perform well with any specific image reconstruction technique.

1.5 Thesis Organization

The rest of the thesis is organized as follows. We investigate a framework for learning unstructured square sparsifying transforms in Chapter 2. The usefulness of the scheme is demonstrated for sparse representation and denoising tasks. Chapter 3 deals with the study of structured square sparsifying transforms. Specifically, we investigate the learning of doubly sparse transforms for signals or image patches. These transforms are a product of a fixed, fast analytic transform such as the DCT, and an adaptive matrix constrained to be sparse. In Chapter 4, we further investigate the methods first introduced in Chapter 2. We develop further the alternating algorithms for learning unstructured (non-sparse) well-conditioned, or orthonormal square sparsifying transforms. A detailed convergence analysis, and the computational advantages, for the new algorithms are presented, along with empirical evidence of their potential in image representation and denoising applications.

In Chapter 5, a novel framework for sparsifying transform-based blind compressed sensing is presented. While compressed sensing (CS) exploits the sparsity of images or image patches in a transform domain or synthesis dictionary to reconstruct images from highly undersampled or compressive measurements, in blind compressed sensing, the underlying sparsifying transform (apart from the image) is also unknown a priori. In Chapter 5, we therefore discuss methods to simultaneously reconstruct the underlying image(s)/volume(s) as well as the square sparsifying transform from highly undersampled measurements. Such a strategy allows greater adaptivity to the measured data. The convergence properties of the proposed algorithms are also described. We then discuss the application of the proposed blind compressed sensing schemes to magnetic resonance imaging. The advantages of the proposed MRI algorithms in terms of image reconstruction quality and runtimes are demonstrated. In Chapter 6, we study the design of data adaptive sampling schemes for compressed sensing MRI. Numerical experiments are presented to demonstrate the advantages of the proposed adaptive sampling method over non-adaptive methods.

Chapter 7 presents a methodology for online learning of square sparsifying transforms. Highly efficient learning algorithms are presented, and demonstrated to be useful in big data applications. In Chapter 8, a rigorous convergence analysis is presented for the online transform learning algorithms. In Chapter 9, we propose a novel convex formulation for doubly sparse square transform learning. Alternative versions of this formulation are also studied. We provide convergence guarantees for the proposed algorithms. Preliminary experiments are also provided to illustrate the usefulness of the proposed convex scheme and its non-convex variants.

Chapters 10 and 11 focus on overcomplete sparsifying transform learning. Chapter 10 investigates a framework for unstructured overcomplete transform learning. The proposed method is shown to be useful in image denoising. In Chapter 11, the focus is on learning structured overcomplete sparsifying transforms, that allow for efficient implementations. Specifically, we propose a union of sparsifying transforms model that is also equivalent to a structured overcomplete sparsifying transform model with block cosparsity, dubbed OCTOBOS. Efficient learning algorithms are presented, along with a convergence analysis, and a demonstration of their usefulness in classification, sparse representation, and denoising tasks. Finally, in Chapter 12, we provide some concluding remarks for the various methods investigated in this work. We also provide suggestions for future directions of study.

CHAPTER 2

LEARNING UNSTRUCTURED SQUARE SPARSIFYING TRANSFORMS

In this chapter, we investigate a framework for square sparsifying transform learning [1]. We refer to the transforms of this chapter as 'unstructured' transforms in order to distinguish them from the (structured) doubly sparse transforms introduced in Chapter 3.

Here, we propose a novel framework for square transform learning that aims to learn well-conditioned sparsifying transforms. We provide efficient algorithms to solve the proposed problems. The learned transforms will be shown to be useful in applications such as signal denoising. In contrast to the prior work on synthesis and analysis dictionary learning (cf. Chapter 1), the transform model allows WY to be approximated by a sparse matrix X, which makes the learning of this model easier. Specifically, with the transform model, sparse coding is cheap and exact. Furthermore, unlike synthesis or analysis dictionary learning, the proposed transform learning formulation in this chapter does not include a highly non-convex function involving the product of two unknown matrices. Moreover, while the convergence analysis in this chapter is only partial, it shows monotone convergence of the objective function in the proposed algorithm. Empirical results demonstrate convergence of the iterates regardless of initial conditions. This desirable convergence behavior contrasts with that of popular synthesis or analysis learning algorithms (e.g., K-SVD), for which no such results, theoretical or empirical, are available.

The rest of the chapter is organized as follows. Our problem formulations for learning sparsifying transforms are described in Section 2.1. Section 2.2 details the proposed algorithms for transform learning and discusses their relevant properties such as convergence and computational cost. In Section 2.3, we introduce a novel signal denoising framework incorporating sparsifying transforms. Section 2.4 demonstrates the performance of our algorithms in-

[1]The material of this chapter has previously appeared in [79, 80, 9].

cluding promising performance of learned transforms in denoising. In Section 2.5, we conclude.

2.1 Transform Learning Problems and Properties

2.1.1 Trivial Solutions, and Full Rank Constraint

Given the training matrix $Y \in \mathbb{R}^{n\times N}$, we propose to minimize the sparsification error given by $\|WY - X\|_F^2$, where $W \in \mathbb{R}^{m\times n}$ and X is column-sparse.

$$\text{(P2.1)} \quad \min_{W,X} \|WY - X\|_F^2 \;\; s.t. \;\; \|X_i\|_0 \leq s \;\; \forall\, i \tag{2.1}$$

The minimization corresponds to fitting the transform model parametrized by W and X to the data Y, by selecting the best (matrix) parameters W and X. The notion of sparsification error can be justified by the fact that natural signals and images can be reasonably approximated in a transform domain (such as Wavelets) using few (here s) significant non-zero transform coefficients. The rest of the coefficients contribute only marginally to the sparsification error.

Problem (P2.1), although intuitive, has a glaring defect. It has a trivial solution $W = 0, X = 0$. In order to avoid the trivial solution, we need to enforce additional constraints or penalties. Introducing a constraint on the norm of W or of its rows (similar to constraints in synthesis [17, 48, 18] or analysis [26, 25] dictionary learning) may seem apt for this setting. However, such a constraint does not preclude the possibility of repeated rows in W. That is, if a particular non-trivial row vector can individually provide the lowest sparsification error for Y (compared to any other row vector), the best solution would be to form W by repeating this row. This problem is exacerbated in the case when Y has rank $n-1$ or lower (i.e., is rank deficient). In this case, all the rows of W can be chosen as scaled versions of a left null vector (left null space has dimension 1 or higher) of Y, leading to a zero sparsification error. However, the resulting W has repeated rows and is not useful.

Therefore, an appropriate constraint must preclude zero rows, repeated rows, or even linearly dependent rows (except when W is a tall matrix, in which case linear dependence of the rows cannot be avoided but may be minimized). In the following, we consider a formulation for the case of square W $(m = n)$.

We propose a full rank constraint to address the ambiguities in the square case. We note that many well-known square sparsifying transforms, such as the DCT, Wavelets, and Hadamard, are non-singular. The one-dimensional finite difference transform, which typically has fewer rows than columns, can be appended with a linearly independent row(s), thereby making it non-singular. The two-dimensional finite difference transform can be obtained as the Kronecker product of two such non-singular (one-dimensional) matrices.

However, full rank is a non-convex and non-smooth constraint. Hence, in the square W setting, we propose to add a negative log-determinant (of W) penalty [81] in the cost. Such a constraint penalizes transforms that have a small determinant (note that full rank is trivially imposed by this constraint).

$$\text{(P2.2)} \quad \min_{W,X} \; \|WY - X\|_F^2 - \lambda \log\det W \quad s.t. \;\; \|X_i\|_0 \leq s \;\; \forall \; i$$

Problem (P2.2) is non-convex. To illustrate this fact, we plot in Figure 2.1, $-\log\det W$ for 2×2 diagonal matrices W as a function of the diagonal elements. The function is well-defined and symmetric in the first and third quadrants (W is positive definite in the first quadrant and negative definite in the third). While the function is convex within each of these quadrants, it is non-convex overall. Note that the domain of the function, which is the union of the first and third quadrants, is also a non-convex set.

One could constrain W in (P2.2) to be positive definite, $W \succ 0$. If in addition the l_0 "norm" for X were relaxed to an l_1 norm, the resulting problem would be jointly convex in W and X. However, enforcing the transform to be positive definite is too restrictive in general. Many well-known sparsifying transforms such as the DCT and Wavelets are not positive definite. Hence, a positive definite constraint will preclude many good candidate transforms from the class of solutions. Therefore, we do not use such a constraint.

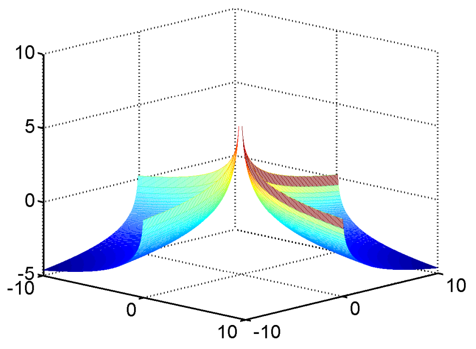


Figure 2.1: Plot of $-\log\det W$ for 2×2 diagonal matrices W. The horizontal axes indicate the first and second diagonal entry values.

2.1.2 Scale Ambiguity and Conditioning

When the data Y admits an exact representation, i.e., there exists a pair $(\tilde{W}, \tilde{X})$ with $\tilde{X}$ sparse such that $\tilde{W}Y = \tilde{X}$, then the cost in (P2.2) can be made arbitrarily small by pre-multiplying $\tilde{W}$ and $\tilde{X}$ by a scalar α (or equivalently, by a diagonal matrix Γ with entries $\pm\alpha$ chosen such that $\det(\Gamma\tilde{W}) > 0$) with $\alpha \to \infty$. The cost becomes unbounded from below in this case, which spells trouble for optimization algorithms. We refer to this phenomenon as the 'scale' ambiguity.

Moreover, the negative log determinant penalty although it enforces full rank, does not necessarily imply good-conditioning. To elucidate the importance of conditioning, consider again the case of Y with rank $n-1$ or lower. Let $\hat{W}$ be a matrix that has the left singular vectors of Y as its rows. We can scale a row of $\hat{W}$ that contains a singular vector corresponding to a zero singular value by $\frac{1}{\epsilon^p}$, where $\epsilon > 0$ is small and p is an integer. All the other rows of $\hat{W}$ can be scaled by ϵ. In this case, such a scaled transform $\hat{W}$ is full rank (has non-zero $\det\hat{W} = \epsilon^{n-1-p}$, which is large for $p \gg n-1$), and provides a sparsification error (computed using the X, whose columns are obtained by thresholding the columns of $\hat{W}Y$) that is close to zero for sufficiently small ϵ. However, this transform is poorly conditioned (condition number $\kappa(\hat{W}) = \frac{1}{\epsilon^{p+1}}$, which is large for large p), and produces only about as much good/non-redundant information as the single row with the large scaling (since the other rows are almost zero in scale). With better conditioning, we would expect the transform to produce more information and less degeneracies. The conditioning of a transform is also an important property for applications such as denoising.

Note that the transform $\hat{W}$ in the aforementioned example also suffers from the scale ambiguity (the objective function decreases without bound as $\epsilon \to 0$). While the example considers an extreme case of rank deficient Y, Problem (P2.2) was also empirically observed to give rise to ill-conditioned transforms for matrices Y constructed from image patches.

In order to address the scale ambiguity and the conditioning of the transform, we introduce an additional norm-constraint $\|W\|_F \leq 1$, which is convex.[2] Alternatively, the constraint can be added as a penalty in the cost to result in the following formulation.

$$\text{(P2.3)} \quad \min_{W,X} \|WY - X\|_F^2 - \lambda \log \det W + \mu \|W\|_F^2 \qquad s.t. \; \|X_i\|_0 \leq s \;\; \forall \; i$$

While we use the l_0 "norm" for sparsity in Problem (P2.3), it could be substituted by a convex l_1 penalty in the cost (i.e., penalize $\|X\|_1 = \sum_{i=1}^N \|X_i\|_1$).

In the following, we will prove that the cost function of Problem (P2.3) is lower bounded and encourages well-conditioning. For convenience, we define

$$Q \triangleq -\log \det W + c \|W\|_F^2 \tag{2.2}$$

Then, the cost function of Problem (P2.3) can be written as follows:

$$\|WY - X\|_F^2 + \lambda Q \tag{2.3}$$

with $c = \frac{\mu}{\lambda}$. Since $\|WY - X\|_F^2 \geq 0$, we proceed to lower bound Q. We define

$$Q_0 \triangleq \frac{n}{2} + \frac{n}{2} \log(2c) \tag{2.4}$$

Lemma 1. *Suppose $W \in \mathbb{R}^{n \times n}$ has positive determinant, and let $Q \triangleq -\log \det W + c \|W\|_F^2$, for some $c > 0$. Then,*

$$Q \geq \frac{n}{2} + \frac{n}{2} \log(2c) - \log \frac{2\kappa}{1 + \kappa^2} \tag{2.5}$$

[2]The Frobenius-norm constraint could be alternatively replaced by constraining each row of W to unit norm. However, this alternative was found empirically to produce inferior transforms than those resulting from Problem (P2.3).

where κ is the 2-norm condition number of W.

Proof. : See Appendix A.1. $\square$

Corollary 1. *$Q \geq Q_0$, with equality if and only if $\kappa = 1$ and the singular values of W are all equal to $\sqrt{\frac{1}{2c}}$.*

Proof. : See Appendix A.2. $\square$

It follows from corollary 1 that the cost function in equation (2.3) is lower bounded by λQ_0. Furthermore, Problem (P2.3) attains this lower bound if and only if there exists a pair $(\hat{W}, \hat{X})$ with $\hat{X}$ sparse (and $\det \hat{W} > 0$) such that $\hat{W}Y = \hat{X}$, and the singular values of $\hat{W}$ are all equal to $\sqrt{0.5\lambda/\mu}$ (hence, the condition number $\kappa(\hat{W}) = 1$). Thus, formulation (P2.3) does not suffer from the scale ambiguity (due to finite lower bound on cost), and favors both a low sparsification error and good conditioning.

Next, we demonstrate that the function Q can be used to derive an upper bound for $\kappa(W)$.

Proposition 1. *Suppose $W \in \mathbb{R}^{n\times n}$ has positive determinant, then*

$$1 \leq \kappa \leq e^{Q-Q_0} + \sqrt{e^{2(Q-Q_0)} - 1}$$

Proof. : Inequality (2.5) in Lemma 1 can be rewritten as $\kappa^2 - 2\kappa e^{Q-Q_0} + 1 \leq 0$. Solving for κ, we get

$$e^{Q-Q_0} - \sqrt{e^{2(Q-Q_0)} - 1} \leq \kappa \leq e^{Q-Q_0} + \sqrt{e^{2(Q-Q_0)} - 1}$$

Since the lower bound above is ≤ 1, we instead use the trivial lower bound of 1 for κ. $\square$

In Proposition 1, the upper bound on κ is a monotonically increasing function of Q. Hence, we conclude that in general, the minimization of the proposed cost function (2.3) (of Problem (P2.3)) encourages reduction of condition number.

Moreover, the following corollary shows that the solution to Problem (P2.3) is perfectly conditioned in the limit of $\lambda \to \infty$. Let the value of Q corresponding to the minimum/optimum value of the cost function (2.3) of Problem (P2.3) be denoted by Q^*, and the condition number of the minimizing transform(s) be denoted by κ^*. We then have the following result.

Corollary 2. *For a fixed $\frac{\mu}{\lambda}$, as $\lambda \to \infty$ in Problem* (P2.3)*, $\kappa^* \to 1$.*

Proof. : By (2.4), for a fixed $c = \frac{\mu}{\lambda}$, Q_0 is fixed too. Suppose we were to minimize the function Q alone, then by corollary 1, the minimum value is Q_0 which is attained by any transform $\hat{W}$ (of positive determinant) that has all of its singular values equal to $\sqrt{\frac{\lambda}{2\mu}}$. Now, as λ is increased in (P2.3) with fixed $\frac{\mu}{\lambda}$, Q^* can only decrease because the Q part of the cost function (2.3) is weighted more heavily. In the limit as $\lambda \to \infty$, we get $Q^* \searrow Q_0$. By proposition 1, we then have that as $Q^* \searrow Q_0$, $\kappa^* \to 1$ (the upper bound on κ^* goes to 1 while the lower bound is 1). $\square$

The upper bound on κ^* decreases as λ is increased with fixed $\frac{\mu}{\lambda}$, suggesting that the minimizing transform(s) can be better conditioned at larger λ. This is also shown empirically in Section 2.4.

Our Problem formulation (P2.3) thus aims to minimize the sparsification error while controlling the condition number and eliminating the scale ambiguity, with the goal of estimating a "good" transform W that provides the best fit to the data. No particular performance metric in an application is directly optimized here. However, as we will show in Sections 2.3 and 2.4, W produced as a solution to Problem (P2.3), either by itself or via its extensions, performs well in applications such as signal representation/recovery, and denoising.

We note that a cost function similar to that in Problem (P2.3), but lacking the $\|W\|_F^2$ penalty (or in other words, similar to (P2.2)) has been derived under certain assumptions in a very different setting of blind source separation [82]. However, the transform learning Problem (P2.3) performs poorly in signal processing applications in the absence of the crucial $\|W\|_F^2$ penalty, which as discussed earlier, helps overcome the scale ambiguity and control the condition number. The superiority of (P2.3) over (P2.2) is also illustrated empirically in Section 2.4.

We also note that penalty terms similar to $-\log\det W$ and $\|W\|_F^2$ (of (P2.3)) can be used to regularize synthesis and analysis dictionary learning in order to enforce full-rank and well-conditioning, and overcome scale ambiguities.

2.1.3 Positive Determinant and Equivalent Solutions

As discussed earlier, Problem (P2.3) is non-convex. Moreover, there is an implicit constraint of $\det W > 0$ for $\log\det W$ to be well-defined. However, this constraint is non-restrictive, because if $(\tilde{W}, \tilde{X})$ is a non-trivial minimizer of the sparsification error term in (P2.3) (or equivalently solves Problem (P2.1)) with $\det \tilde{W} < 0$, then we can always form an equivalent pair $(\Gamma\tilde{W}, \Gamma\tilde{X})$ (where Γ is a diagonal "sign matrix" with ± 1 on the diagonal and $\det \Gamma < 0$), which provides the same sparsification error, but has $\det \Gamma\tilde{W} > 0$. Therefore, it suffices to only consider transforms that have $\det W > 0$.

Furthermore, the $\det W > 0$ constraint need not be enforced explicitly. This is because the cost function in (P2.3) has log-barriers (see Figure 2.1) in the space of matrices at W for which the determinant is less than or equal to zero. These log-barriers help prevent an iterative minimization algorithm initialized with W satisfying $\det W > 0$ from getting into the infeasible regions, where $\det W \leq 0$.

The problem has one remaining inherent ambiguity. Similar to synthesis dictionary learning [48], Problem (P2.3) admits an equivalence class of solutions/minimizers. Given a particular minimizer $(\tilde{W}, \tilde{X})$, we can form equivalent minimizers by simultaneously permuting the rows of $\tilde{W}$ and $\tilde{X}$ (only permutations that retain the sign of the determinant of $\tilde{W}$ are permitted). Pre-multiplying a minimizer by a diagonal sign matrix Γ with $\det \Gamma > 0$ also provides an equivalent minimizer [3]. This ambiguity between completely equivalent solutions is of no concern, and we do not attempt to eliminate it. Which of the equivalent solutions will be obtained by an iterative algorithm for the solution of (P2.3) will depend on initialization.

2.2 Algorithm and Properties

2.2.1 Algorithm

Our algorithm for solving Problem (P2.3) alternates between updating X and W.

[3] We provide a more general characterization of the set of equivalent minimizers in transform learning later in Section 4.2 of Chapter 4.

2.2.1.1 Sparse Coding Step

In this step, we solve Problem (P2.3) with fixed W.

$$\min_{X} \|WY - X\|_F^2 \quad s.t. \quad \|X_i\|_0 \leq s \ \forall \ i \tag{2.6}$$

The solution X can be computed exactly by thresholding WY, and retaining only the s largest coefficients (magnitude-wise) in each column. Contrast this with the sparse coding step in synthesis dictionary learning [17], which can only be solved approximately using techniques such as OMP. If instead the l_0 "norm" is relaxed to an l_1 norm and added as a penalty in the cost, we solve the following problem.

$$\min_{X} \|WY - X\|_F^2 + \eta \sum_{i=1}^{N} \|X_i\|_1 \tag{2.7}$$

The solution for X in this case can again be exactly computed by soft thresholding as follows.

$$X_{ij} = \begin{cases} (WY)_{ij} - \frac{\eta}{2} & , \ (WY)_{ij} \geq \frac{\eta}{2} \\ (WY)_{ij} + \frac{\eta}{2} & , \ (WY)_{ij} < -\frac{\eta}{2} \\ 0 & , \ else \end{cases} \tag{2.8}$$

Here, subscript ij indexes matrix entries. We will show that such soft thresholding is even cheaper than the projection onto the l_0 ball (2.6) by hard thresholding.

2.2.1.2 Transform Update Step

In this step, we solve Problem (P2.3) with fixed X. This involves the unconstrained minimization

$$\min_{W} \|WY - X\|_F^2 - \lambda \log \det W + \mu \|W\|_F^2 \tag{2.9}$$

This problem can be solved using methods such as steepest descent, or conjugate gradients. We can employ the conjugate gradient method with backtracking line search [83] (also known as Armijo rule), which typically converges faster than steepest descent. Fixed step size rules were also observed to work well and faster in practice.

The gradient expressions for the various terms in the cost [84] are as follows. We assume that $\det W > 0$ on some neighborhood of W, otherwise log() would be discontinuous.

$$\nabla_W \log\det W = W^{-T} \tag{2.10}$$

$$\nabla_W \|W\|_F^2 = 2W \tag{2.11}$$

$$\nabla_W \|WY - X\|_F^2 = 2WYY^T - 2XY^T \tag{2.12}$$

Various stopping rules can be used for the conjugate gradient iterations, such as the norm of the gradient of the objective function dropping below a threshold. However, the conjugate gradient algorithm typically converges quickly, and a fixed number of iterations were empirically observed to work well.

As described above, the update of W is performed with fixed X. Alternative strategies such as updating W jointly with the non-zero values in X (using the conjugate gradient method) for a fixed sparsity pattern of X (similarly to K-SVD), yielded similar empirical performance, albeit at a considerably higher computational cost.

For the proposed alternating algorithm to work, W must be initialized to have a positive determinant. The alternating algorithm itself typically converges quickly, and as shown in Section 2.4, a fixed number of iterations suffices in practice.

2.2.2 Convergence

The algorithm for solving Problem (P2.3) alternates between sparse coding and transform update steps. The solution for the sparse coding step is exact/analytical. Thus, the cost function can only decrease in this step. For the transform update step, the solution is obtained by conjugate gradients (for instance with Armijo step size rule). Thus, in this step too, the cost function can again only decrease. The cost function being monotone decreasing and lower bounded, it must converge. While convergence of the iterates themselves for the proposed alternating minimization of the non-convex problems does not follow from this argument, the iterates are found empirically to converge as well.

2.2.3 Computational Cost

The algorithm for Problem (P2.3) involves Sparse coding steps and Transform Update steps. In the sparse coding step, when the ℓ_0 "norm" is used for sparsity of X, then the projection onto the ℓ_0 ball by hard thresholding, if done by full sorting [85], involves $O(n \log n)$ comparisons per training signal, or a total of $O(nN \log n)$ operations. Using instead the ℓ_1 norm penalty for sparsity, the soft thresholding in (2.8) requires only $O(nN)$ operations, and is therefore cheaper than projecting onto the ℓ_0 ball. Either way, computing WY (prior to thresholding) requires $O(Nn^2)$ operations, and dominates the computation in the sparse coding step.

To estimate the cost of the Transform Update step, assume that YY^T has been pre-computed (at a total cost of $O(Nn^2)$ for the entire algorithm). The gradient evaluation in equation (2.12) involves the matrix products WYY^T and XY^T. The product XY^T is computed once per Transform Update step. Computing $W(YY^T)$ requires n^3 multiply-add operations. Furthermore, when X is sparse with Ns non-zero elements and $s = \alpha n$ (where typically $\alpha \ll 1$), then computing the product XY^T requires αNn^2 multiply-add operations. (Note that when the ℓ_1 norm is used for sparsity of X, one would need to carefully choose the parameter η in (2.8), to get the αNn^2 cost for computing XY^T.) Next, the computation for gradient evaluations in (2.10) and (2.11) is dominated by $C_3 n^3$ (for the matrix inverse), where C_3 is a constant. Thus, the transform update step has computational cost of roughly $\alpha Nn^2 + (1 + C_3)Ln^3$, where L is the number of conjugate gradient steps. Assuming $(1 + C_3)nL < \alpha N$, the cost per Transform Update step scales as $O(n^2 N)$.

The total cost per iteration (of Sparse coding and Transform Update) of the algorithms thus scales as $O(Nn^2)$. Contrast this with the cost per iteration of the synthesis dictionary learning algorithm K-SVD [17], which roughly scales as $O(sNn^2)$ (cost dominated by the sparse coding step) for the square dictionary case [1, 5]. Since $s = \alpha n$, the K-SVD cost scales as $O(Nn^3)$. The cost per iteration of the analysis K-SVD algorithm [23, 26] also scales similarly. Our transform-based algorithm thus provides a reduction of the computational cost relative to synthesis/analysis K-SVD in the order, by factor n. Computation times shown in the next section confirm the significant speed-ups of transform learning over K-SVD.

2.3 Application to Signal Denoising

We introduce a novel problem formulation for signal denoising using adaptive sparsifying transforms. In this application, we are given N data signals/vectors arranged as columns of matrix $Y \in \mathbb{R}^{n \times N}$ that are corrupted by noise, i.e., $Y = Y^* + H$, where Y^* are the original noiseless data and H is the corrupting noise (a matrix). The goal is to estimate Y^* from Y. We propose the following formulation for denoising signals using learned sparsifying transforms:

$$\text{(P2.4)} \quad \min_{W,X,\hat{Y}} \left\| W\hat{Y} - X \right\|_F^2 + \lambda Q(W) + \tau \left\| Y - \hat{Y} \right\|_F^2 \qquad (2.13)$$
$$s.t. \ \ \|X_i\|_0 \leq s \ \ \forall \ i$$

where the functional Q was defined in Section 2.1 (with $c = \mu/\lambda$), and represents the portion of the cost depending only on W. The formulation assumes that the noisy Y can be approximated by $\hat{Y}$ that is approximately sparsifiable by a learned sparsifying transform W (i.e., $W\hat{Y} \approx X$, with X being column-sparse). This assumption for noisy signals can also be viewed as a generalization of the transform model to allow for explicit denoising. The assumptions $Y \approx \hat{Y}$, $W\hat{Y} \approx X$, with X being column-sparse, can then be regarded as a *"noisy signal" version* of the transform model [4].

The parameter τ in Problem (P2.4) is typically inversely proportional to the noise level σ [1]. When $\sigma = 0$, the optimal $\hat{Y} = Y = Y^*$, and Problem (P2.4) reduces to Problem (P2.3). Note that while (P2.4) is aimed at explicit denoising, Problem (P2.3) is aimed at signal representation, where explicit denoising may not be required.

Problem formulation (P2.4) for simultaneously solving W, X, and $\hat{Y}$ is non-convex even when the l_0 "norm" is relaxed to an l_1 norm. We propose a simple and fast alternating algorithm to solve Problem (P2.4). In one step of

[4]We do not explore the usefulness of Problem (P2.4) with fixed W in this work, focusing instead on adaptivity.

this alternating algorithm, $\hat{Y}$ and W are fixed (in (P2.4)), and X is obtained by thresholding $W\hat{Y}$ (same as the sparse coding step in the algorithm for (P2.3)). In the second step, $\hat{Y}$ and X are fixed (in (P2.4)), and W is updated using the conjugate gradient method (same as the Transform Update step in the algorithm for (P2.3)). In the third step, W and X are fixed (in (P2.4)), and $\hat{Y}$ is updated by solving the following simple least squares problem.

$$\min_{\hat{Y}} \left\| W\hat{Y} - X \right\|_F^2 + \tau \left\| Y - \hat{Y} \right\|_F^2 \tag{2.14}$$

The promise of the proposed denoising framework will be demonstrated in the next section.

2.4 Numerical Experiments

We demonstrate the promise of our adaptive formulations in terms of their ability to provide low sparsification errors, good denoising, etc. While the algorithms can be initialized with various useful/relevant transforms, such initializations must have positive determinant (or else we can swap two rows to ensure positive determinant). We work with the l_0 "norm" for sparsity of X in the experiments, but this can be easily substituted by an l_1 norm penalty. We use a fixed step size in the transform update step of our algorithms here. Backtracking line search (employing the Armijo search rule) gives similar performance, but is slower. All implementations were coded in Matlab v7.8 (R2009a). Our algorithm is compared to the synthesis K-SVD [17] using the implementation available from Michael Elad's website [86]. Computations were performed with an Intel Core i5 CPU at 2.27GHz and 4GB memory, employing a 64-bit Windows 7 operating system.

We first demonstrate the ability of our formulation (P2.3) to learn a sparsifying transform. We consider both synthetic and real data in our experiments. Synthetic data are generated (without noise) as sparse linear combinations of the atoms of a square random (full rank) synthesis dictionary D [17]. Such data are also exactly sparsifiable by D^{-1}. Real data are generated as non-overlapping patches of natural images. Since real data are not exactly sparsifiable, the transform model is well-suited to such data. For this case, we compare the sparsification errors of transforms learned using our algorithms with the corresponding errors of analytical transforms. The data, both synthetic and real, will be used to demonstrate the properties of our algorithm such as convergence, well-conditioned final transform, insensitivity to initialization, etc. Finally, we illustrate the promise of our signal denoising formulation and algorithm through simple experiments.

The quality of the learned transforms in our experiments will be judged based on their condition number and sparsification error. We will argue and illustrate in this section that the performance of the learned transform in applications depends on the trade-off between the sparsification error and condition number.

We also define the 'normalized sparsification error' as $\|WY - X\|_F^2 / \|WY\|_F^2$. This measures the fraction of energy lost in sparse fitting in the transform domain. In other words, it indicates the degree of energy compaction achieved in the transform domain, or how well the transform model holds for the signals. This is an interesting property to observe for the adaptive transforms.

For images, another useful metric is the recovery peak signal to noise ratio (or *recovery PSNR*) defined (in dB) as the scaled (by the factor 20) base-10 logarithm of $255\sqrt{P}/\|Y - W^{-1}X\|_F$, where P is the number of image pixels. This measures the error in recovering the patches Y (or equivalently, the image in the case of non-overlapping patches) as $W^{-1}X$ from their sparse codes X obtained by thresholding WY. While we consider here the option of image recovery from the sparse code, the sparse representations are in general used differently in various applications such as denoising.

2.4.1 Sparsification of Synthetic Data

2.4.1.1 Case 1 - Generating Transform Not Unit-Conditioned

We generate a synthetic 20×20 synthesis dictionary with zero mean and unit variance i.i.d. Gaussian entries. The data matrix Y has 200 training signals each of which is generated as linear combination of a random set of $s = 4$ atoms of the synthetic dictionary. The coefficients in the linear combination are chosen to be zero mean and unit variance i.i.d. Gaussian random variables. Problem (P2.3) is solved with parameters $\lambda = 50$, $\mu = 10^{-4}$, to

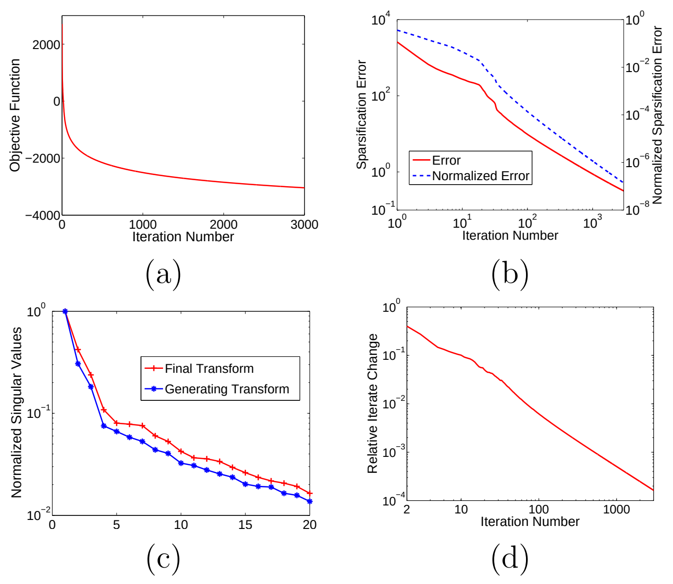


Figure 2.2: Algorithm with synthetic data: (a) Objective function vs. iterations, (b) Sparsification error and normalized sparsification error vs. iterations, (c) Normalized singular values of final and generating transforms, (d) Relative iterate change $\|W_i - W_{i-1}\|_F \,/\, \|W_{i-1}\|_F$ vs. iterations beginning with iteration 2.

learn a transform W that is adapted to the data. The initial transform is the identity matrix. In each transform update step, the conjugate gradient algorithm was run for 60 iterations with a fixed step size of 10^{-4} (although we use more CG iterations here, even 30 or fewer iterations suffice typically).

Figure 2.2 shows the progress of the algorithm over iterations. The objective function of (P2.3) (Figure 2.2(a)) converges monotonically over the iterations (cannot decrease below the bound λQ_0 derived in Section 2.1 which is -5.7146×10^3 for this case). The sparsification error and normalized sparsification error (Figure 2.2(b)) also decrease/converge quickly. The normalized error decreases below 0.01 (or 1%) by the 23^{rd} iteration and below 0.001 by the 44^{th} iteration. Thus, a small number of algorithm iterations seem to suffice to reach a good sparsification error.

Figure 2.2(c) shows the normalized singular values (normalized by the largest singular value) of the final learned transform, and generating transform (i.e., inverse of the synthetic dictionary used to generate the data) on a log-scale. The condition number of the learned transform is 60.8, while that of the generating transform is 73. This indicates the ability of our algorithm to converge to well-conditioned (not just full rank) transforms. Moreover, the algorithm provides better conditioning compared to even the generating transform/dictionary. In Figure 2.2(d), we plot the relative change between successive iterates/transforms defined as $\|W_i - W_{i-1}\|_F \,/\, \|W_{i-1}\|_F$, where i denotes the iteration number. This quantity decreases to a low value of 10^{-4} over the iterations, indicating convergence of the iterates.

Since the synthetic data are exactly sparsifiable (i.e., they satisfy the transform model $WY = X + E$ with $E = 0$, or equivalently, they satisfy the analysis model since WY is column-sparse), the results of Figure 2.2 indicate that when the analysis model itself holds, our transform learning algorithm for (P2.3) can easily and cheaply find it.

2.4.1.2 Case 2 - Unit-Conditioned Generating Transform

We now consider the case when the generating dictionary has equal singular values and evaluate the performance of the algorithm for this case. The goal is to investigate whether the algorithm can achieve the lower bounds derived in Section 2.1.

For this experiment, a synthetic 20×20 dictionary with i.i.d. Gaussian entries is generated and its singular value decomposition (SVD) of form $U\Sigma V^H$ is computed. An orthonormal dictionary is then generated as UV^H. The data Y are generated using this dictionary similarly to the experiment of Figure 2.2. The transform learning algorithm parameters are set as $\lambda = 0.5$, $\mu = 0.25$. The initial transform is another random matrix with i.i.d. Gaussian entries. All other algorithm parameters are set similarly to the experiment of Figure 2.2.

The objective function, sparsification error, and condition number (of W) are plotted over iterations for our algorithm (for (P2.3)) in Figures 2.3(a) and 2.3(b), respectively. In this case, the objective function converges to a value of 5 which is the lower bound (λQ_0) predicted in Section 2.1. The sparsification error and condition number converge to 0 and 1 respectively. The normalized sparsification error (not shown here) drops below 0.01 (or 1%) by the 14th iteration. Thus, the learned transform provides zero sparsification error, and has equal singular values as predicted in Section 2.1 (when the lower bound of the objective is achieved). Moreover, since $\sqrt{\frac{\lambda}{2\mu}} = 1$ (by choice of parameters), the singular values of the learned transform turn out to be all equal to 1 as predicted in Section 2.1. Thus, when a transform that satisfies the lower bound of the objective function exists, the algorithm converges to it.

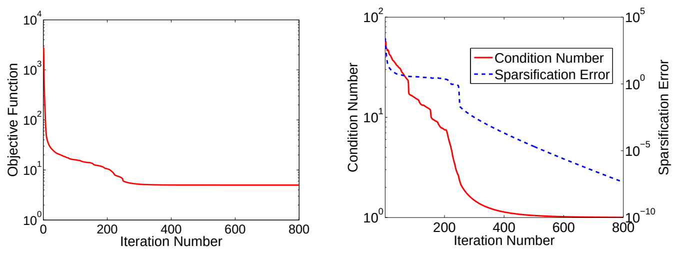
Note that when the data are generated using a synthetic dictionary with condition number $\kappa_{gen} > 1$, the algorithm tends to produce a transform $\hat{W}$ with condition number $\kappa(\hat{W}) < \kappa_{gen}$ lower than that of the generating dictionary. A condition number $\kappa(\hat{W}) = 1$ may not be achieved, since a transform that gives both low sparsification error and a condition number of 1 may not exist in this case. This was also observed in the experiment of Figure 2.2.

2.4.2 Sparsification of Real Data

2.4.2.1 Insensitivity to Initialization

Here, we extract the 8×8 ($n = 64$) non-overlapping patches from the image Barbara [17]. The data matrix Y in this case has 4096 training signals (patches represented as vectors) and we work with $s = 11$. The means (or DC values) of the patches are removed and we only sparsify the mean-subtracted patches (mean removal is typically adopted in image processing applications such as K-SVD based image denoising). The means can be added back for display purposes. Problem (P2.3) is solved to learn a square transform W that is adapted to this data. The algorithm parameters are $\lambda = \mu = 4 \times 10^5$. The conjugate gradient algorithm was run for 128 iterations in each transform update step, with a fixed step size of 10^{-8}. (The performance is similar even with 20 or less conjugate gradient iterations.)

We consider four different initializations (initial transforms) for the algorithm. The first is the 64×64 2D DCT matrix (defined as $W_0 \otimes W_0$, where W_0 is the 8×8 1D DCT matrix, and "$\otimes$" denotes the Kronecker product). The second initialization is obtained by inverting/transposing the left singular matrix of Y. This initialization is also popularly known as the Karhunen-Loève Transform (KLT). The third and fourth initializations are the identity matrix, and a random matrix with i.i.d. Gaussian entries (zero mean and standard deviation 0.2), respectively. Note that any initial matrix with a negative determinant can be made to have a positive determinant by



(a) (b)

Figure 2.3: Algorithm attains lower bound: (a) Objective function vs. iterations, (b) Condition number and sparsification error vs. iterations.

switching the signs of the entries on one row, or by exchanging two rows.

Figure 2.4(a) shows the objective function of Problem (P2.3) over the iterations of the algorithm for the different initializations. The objective function converges monotonically, and although different initializations lead to different initial rates of convergence, the objective functions have nearly identical final values in all cases. This indicates that our alternating algorithm is robust to initialization. The sparsification error (Figures 2.4(b) and 2.4(d)) too converges quickly and to similar values for all initializations. The horizontal lines in Figures 2.4(b) and 2.4(d) denote the sparsification errors of the 2D DCT, KLT, identity, and random Gaussian transforms (i.e., the sparsification error at iteration zero). Our algorithm reduces the sparsification error by 5.98 dB, 7.14 dB, 14.77 dB, and 18.72 dB, respectively, from the values for the initial transforms. The normalized sparsification error for the learned transform W with the 2D DCT initialization is 0.0437. The values corresponding to the other initializations are only slightly different.

The recovery PSNR for the learned W is 34.59 dB with the 2D DCT initialization. The corresponding values for the other initializations differ only by hundredths of a dB. (These small gaps could be reduced further with better choice of step size and the number of conjugate gradient iterations.)

Finally, the condition number (Figure 2.4(c)) with the random Gaussian initialization also converges quickly to a low value of 1.46. This illustrates that image patches are well sparsified by well-conditioned transforms.

To study further the effect of different initializations, Figure 2.4(e) compares the singular values of the transforms learned with 2D DCT and with random Gaussian initializations. The singular values for the two cases are

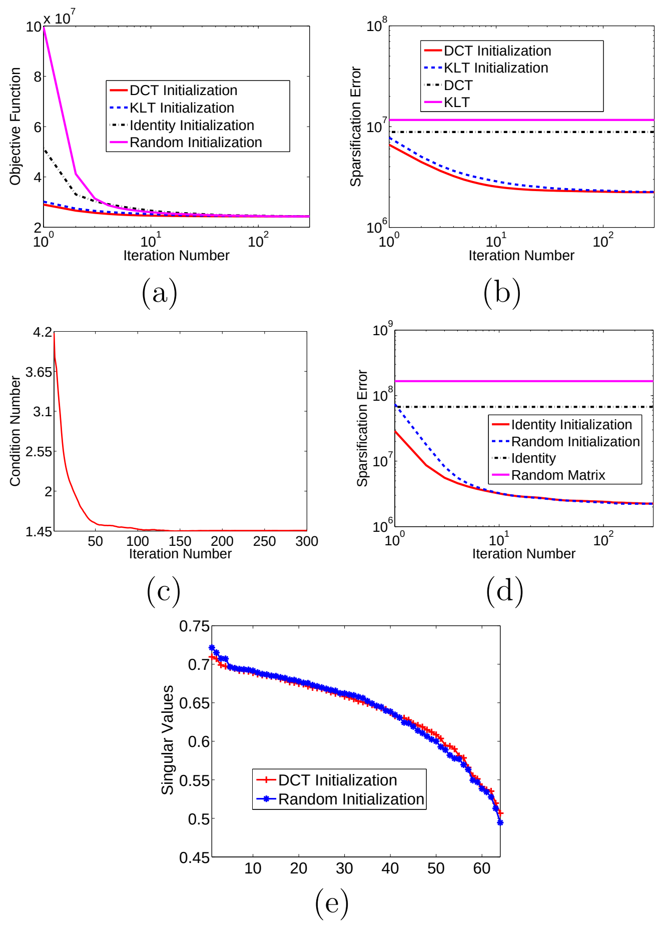


Figure 2.4: Real data - Effect of different Initializations: (a) Objective function vs. iterations, (b) Sparsification error vs. iterations for DCT and KLT initializations, along with the sparsification errors (horizontal lines) of the DCT and KLT transforms, (c) Condition number vs. iterations for random Gaussian initialization, (d) Sparsification error vs. iterations for identity and random Gaussian initializations, along with the sparsification errors (horizontal lines) of the identity and random Gaussian matrices themselves, (e) Singular values of the transforms learned with DCT and random Gaussian initializations.

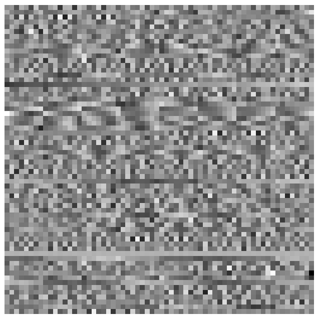
almost identical, with condition numbers of 1.40 and 1.46, respectively.

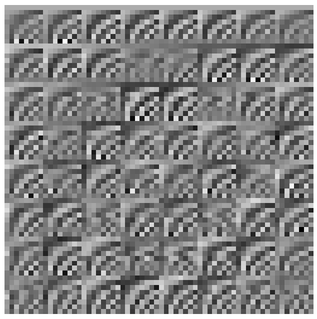
For direct comparison, the transforms learned with the random Gaussian and DCT initializations are shown in Figures 2.5(a) and 2.5(b), respectively. Figures 2.5(c) and 2.5(d) provide a different view of the learned transforms, with each row of (learned) W displayed as an 8×8 patch, which we call the 'transform atom'. Figure 2.5(f) is obtained from Figure 2.5(c) by rearranging (or, permuting) the transform atoms [5]. We see that the transforms learned with the DCT and random Gaussian initializations have many similarities, but also some differences (observed by comparing corresponding atoms in Figures 2.5(d) and 2.5(f)). Figure 2.6(a) shows the magnitude of the cross-gram matrix [6] computed between the transforms in Figures 2.5(f) and 2.5(d). We normalize the rows of the transforms prior to computing their cross-gram matrix. The cross-gram matrix then indicates the coherence between every pair of rows from the two learned transforms. For the 64×64 cross-gram matrix in this case, there are only 30 entries with amplitude above 0.9 and 52 entries with amplitude above 0.7. This indicates that the transforms learned with the DCT and random Gaussian initializations are not related by only row permutations and sign changes.

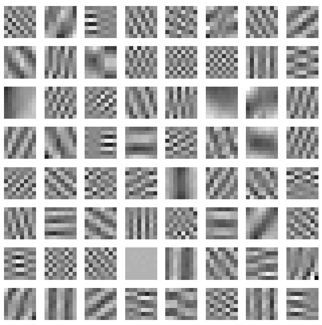
While the two learned (with different initializations) transforms appear somewhat different, they are essentially equivalent in the sense that they produce very similar objective function values (in (P2.3)), sparsification errors, and have almost identical singular values (and thus, almost identical condition numbers). In both cases, the atoms exhibit geometric and frequency like structures. Apparently, the transform learning algorithm is able to discover the structure of image patches (even with a random initialization), and provide dramatically lower sparsification errors than analytical transforms. In particular, the atoms of the learned transforms display some distinctions from the those of the analytical patch-based (orthonormal) 2D DCT (Figure 2.5(e)). For example, some of the learned atoms show oriented textures (adapted for the image Barbara), that are not captured by the DCT. For further insight, we also show in Figure 2.6(b) the magnitude of the 64×64 cross-gram matrix computed between the row-normalized version of the learned transform in Figure 2.5(f) and the 2D DCT. In this case, there are only 13 entries in Figure 2.6(b) with amplitude above 0.9, indicating that the learned transform has many features dissimilar from the DCT. Alternatively, when the 64×64 cross-gram matrix is computed between the row-normalized version of the learned transform in Figure 2.5(d) and the 2D DCT, the result has only 18 entries with magnitude above 0.9, indicating

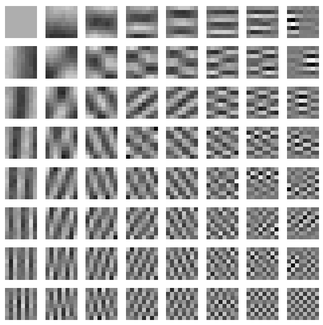
[5]The atoms in Figure 2.5(c) were rearranged to make the result appear more like Figure 2.5(d).

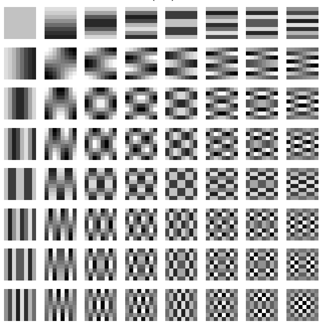
[6]For two matrices A and B of the same size, the cross-gram matrix is computed as AB^T.


(a)


(b)


(c)


(d)


(e)

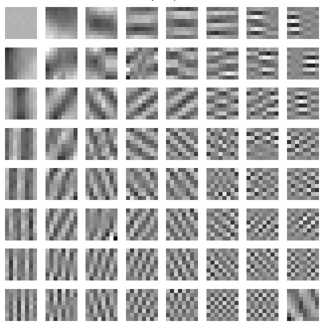

(f)

Figure 2.5: Real data - Effect of different Initializations: (a) Transform learned with random Gaussian initialization, (b) Transform learned with DCT initialization, (c) Rows of learned transform shown as patches for the case of random Gaussian initialization, (d) Rows of learned transform shown as patches for DCT initialization, (e) Rows of the (patch-based) 2D DCT W shown as patches, (f) Learned transform in Figure 2.5(c) with its rows (that are shown as patches) rearranged.

again that the learned transform (even with DCT initialization) has many features dissimilar from the DCT.

Based on the results of Figures 2.4, 2.5, and 2.6, we conjecture that in general, Problem (P2.3) may admit multiple global minima that (potentially) may not be related by only row permutations and sign changes. Which of

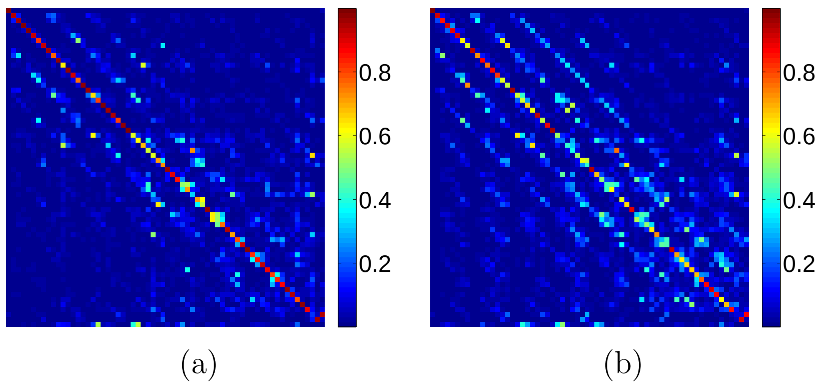


Figure 2.6: Real Data Case. Magnitude of the cross-gram matrix computed: (a) between the row-normalized versions of the learned transforms in Figures 2.5(f) and 2.5(d); and (b) between the row-normalized version of the learned transform in Figure 2.5(f) and the DCT in Figure 2.5(e).

these essentially equivalent solutions is actually achieved by the algorithm depends on the initialization. The existence of alternative but essentially equivalent solutions to (P2.3) also suggests that additional application-specific performance criteria may be used to select between equivalent transforms, or the problem formulation may be modified to incorporate additional preferences.

2.4.2.2 Performance for Various Images vs. DCT

Next, we study the behavior of our algorithm on different images, by solving (P2.3) to learn a transform for each of four different 512×512 images. The transforms are directly adapted to the non-overlapping patches of the images. All algorithm parameters are the same as for the experiment of Figure 2.4 (and we use the DCT initialization).

Table 2.1 lists the normalized sparsification errors and recovery PSNRs for the learned transforms and the patch-based 2D DCT (at $s = 11$), along with the condition numbers of the learned transforms. The learned transforms are seen to be well-conditioned for all the images. The corresponding normalized sparsification errors are small and, moreover, better than those of the 2D DCT by up to 3.4 dB for the tested images. The learned transforms also provide up to 3.1 dB better recovery PSNRs than the 2D DCT. All these results indicate the promise of the adaptive transform model for natural signals.

	CN-L	NSE-L	rPSNR-L	NSE-D	rPSNR-D
Barbara	1.40	0.0437	34.59	0.0676	32.85
Lena	1.16	0.0376	37.64	0.0474	36.91
Peppers	1.17	0.0343	36.97	0.0448	36.15
Cameraman	1.13	0.0088	42.50	0.0191	39.43

Table 2.1: Normalized sparsification errors (NSE-L) and recovery PSNRs (rPSNR-L) for the learned transforms, along with the corresponding values for the 2D DCT (NSE-D/rPSNR-D), and the condition numbers of the learned transforms (CN-L).

Note that while we considered adapting the transform to specific images, a transform adapted to a data-set of training images also gives promising improvements over fixed transforms such as the DCT and Wavelets on test images – a property that can be exploited for image compression.

2.4.3 Performance as Function of Parameters

Next, we work with the same data as for Figure 2.4, and test the performance of our algorithm as a function of the parameter λ at different data sparsity levels $s = 7, 11, 15$ (with $\mu = \lambda$, and all other parameters fixed as in the experiment of Figure 2.4). The algorithm is initialized with the 2D DCT for the experiments.

For fixed s, the condition number of the learned transform (Figure 2.7(b)) decreases as a function of λ. For high values ($> 10^6$) of λ (and $\mu = \lambda$), the algorithm favors a condition number of 1. For low values of λ, the condition number is quite high (condition number of 70 at $\lambda = 10^3$ when $s = 11$). This behavior of the condition number for fixed $\frac{\mu}{\lambda}$, was predicted in Section 2.1.

On the other hand, the normalized sparsification error (Figure 2.7(a)) increases with λ for fixed s. The error is lowest for smaller λ. This is because smaller λ values give higher preference to the sparsification error term in the cost (2.3) of Problem (P2.3). Moreover, even at high values of λ, the normalized sparsification error tends to be reasonable. (For example, it is 0.0457 at $\lambda = 10^6$ when $s = 11$.) Very high λ values tend to increase the normalized sparsification error only slightly. The plots indicate that we can get reasonable normalized sparsification errors simultaneously with good condition numbers.

Figure 2.7(c) plots the recovery PSNRs with the learned transforms for Barbara. For fixed s, the recovery PSNR is best at λ values corresponding to intermediate conditioning or 'well-conditioning'. (For example when $s = 11$, the best recovery PSNR is 34.65 dB at $\lambda = 10^5$, or at $\kappa = 2.29$.) At unit conditioning, or bad conditioning, the recovery PSNR is lower. This indicates that natural images tend to prefer well-conditioned transforms, as far as recovery PSNR is concerned.

All the metrics, however, degrade when the data sparsity level s is reduced (for fixed λ, μ). This behavior is expected since at lower s values, we have fewer degrees of freedom to represent the data (i.e., the learning is more constrained at lower data sparsity levels). For a particular choice of condition number (from Figure 2.7 (b)), we get a lower normalized sparsification error at a larger value of s, and vice-versa.

As the data sparsity level $s \nearrow n$ (where n is the number of pixels in a patch), we can expect all the metrics to improve, because the problem becomes less and less constrained. In the limit when $s = n$, we can have $WY = X$ exactly and thus, infinite recovery PSNR with orthonormal transforms such as the DCT, or even with the trivial identity matrix/transform. Thus, when $s = n$, Problem (P2.3) attains the lower bound (with any W satisfying Corollary 1) of its cost function.

The normalized sparsification errors for the (patch-based) 2D DCT at $s = 7, 11, 15$ are 0.1262, 0.0676, and 0.0393, respectively. The corresponding values for the recovery PSNR are 30.14 dB, 32.85 dB, and 35.21 dB, respectively. At reasonable condition numbers, the learned transforms at $s = 7, 11, 15$ perform much better than the DCT at those sparsity levels.

Thus, it is evident from the results of Figure 2.7 that the parameters can provide a trade-off between the various metrics such as sparsification error and condition number. The choice of parameters would also depend on the specific application (i.e., on how sparsification error and condition number affect the performance in the specific application). For example, when the goal is recovery from sparse code, we see that since the transform model suggests $WY = X + E$, where E is the sparsification error term in the transform domain, we have $Y = W^{-1}X + W^{-1}E$, when W is non-singular. The recovery PSNR depends on the quantity $W^{-1}E$, which in turn depends on both the sparsification error and conditioning of W. Thus, the trade-off between sparsification error and condition number can be expected to determine the

Sparsity	1	5	10	15	20	25	30	35	40	45	50	55	60
rPSNR-L	24.4	30.2	34.0	36.9	39.6	42.1	44.6	47.3	50.4	54.1	58.7	65.5	78.6
rPSNR-D	23.8	28.5	32.2	35.2	37.9	40.5	43.0	45.7	48.6	51.9	56.1	62.1	72.8

Table 2.2: Recovery PSNRs for the learned transforms (rPSNR-L), along with the corresponding values for the 2D DCT (rPSNR-D) at various sparsity levels (s) for the Barbara image.

best recovery PSNR. Therefore, although the formulation of Problem (P2.3) does not directly optimize for the recovery PSNR, it attempts to do so indirectly by allowing control over the sparsification error and condition number, which in turn serve as good surrogates for the recovery PSNR.

Table 2.2 shows the recovery PSNRs for the learned transforms (the data and all parameters except s are fixed as in the experiment of Figure 2.4 and the learning is initialized with the 2D DCT) and the DCT at several sparsity levels (s). The learned transforms perform better than the DCT at all the sparsity levels in Table 2.2, and provide roughly 2 dB improvement in rPSNR over the DCT at most sparsity levels.

An interesting point to note is that since our algorithm can learn well-conditioned transforms, the quantities $\|WY - X\|_F^2$ and $\|Y - W^{-1}X\|_F^2$ are both well-behaved. In contrast, K-SVD [17] typically provides very poorly conditioned dictionaries D. The term $\|D^{-1}Y - X\|_F^2$ is typically too high for K-SVD (even much worse than the sparsification errors of analytical transforms). Thus, the inverse of the K-SVD dictionary may hardly qualify as a sparsifying transform.

Note that setting $\frac{\mu}{\lambda} = 1$ (with λ chosen appropriately) worked well for most experiments except for synthetic data cases with poor generating conditioning for which the parameters were set manually.

Next, in order to illustrate the importance of the Frobenius-norm regularization in our formulation (P2.3), we repeat the experiment of Figure 2.4 (with DCT initialization), but with $\mu = 0$ (or, in other words, we solve Problem (P2.2)). In this case, the constant atom (row of all 1's) is an exact sparsifier (or, orthogonal) for the zero-mean image patches. Thus, a transform (of positive determinant) that has a constant row scaled by $\alpha \to \infty$, and some other linearly independent rows scaled by $\hat{\alpha} \to 0$, would be a good sparsifier as per Problem (P2.2). The Frobenius-norm term in the cost of (P2.3) is necessary to overcome such scaling ambiguities, and for encouraging well-conditioning. Hence, when our algorithm is executed for many

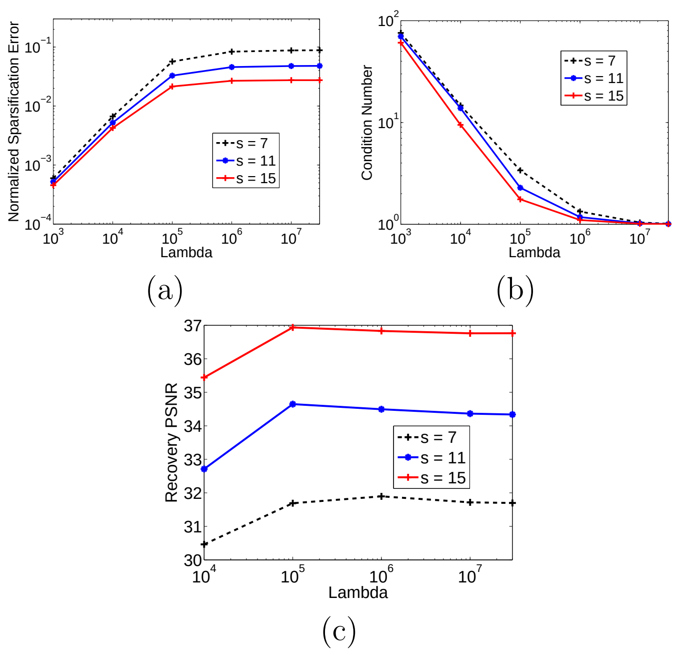


Figure 2.7: Parameter Selection: (a) Normalized sparsification error vs. λ for different values of s with $\mu = \lambda$, (b) Condition number vs. λ for different s values with $\mu = \lambda$, (c) Recovery PSNR vs. λ for different values of s with $\mu = \lambda$.

iterations (5000 iterations) with $\mu = 0$, we obtain a badly conditioned transform, whose condition number is $\kappa = 1037$. The learned transform also has a high Frobenius norm (of 460) which is mostly concentrated on the constant atom that is learned by our algorithm. (Note that a few other rows of the learned W also have high norms, but the constant row has a much higher norm.) The normalized sparsification error and recovery PSNR for this case (after 5000 iterations) are 0.003 and 32.59 dB, respectively. The low value for the normalized sparsification error is expected, since the lack of the Frobenius-norm regularization term in (P2.3) favors better sparsification. However, the recovery PSNR is much worse (by 2 dB) than that obtained in the experiment of Figure 2.4, due to the poor conditioning of the learned W [7]. Thus, the Frobenius-norm regularization is crucial for removing

[7]Note that, although the condition number obtained with $\mu = 0$ (after 5000 iterations) is substantially higher than that obtained with $\mu = \lambda$, the drop in the recovery PSNR is not as significant. This is because, the high norm of the constant row has no effect on the sparsification error term $E = WY - X$. Hence, the quantity $W^{-1}E$ (which determines the recovery PSNR), although degraded by the poor conditioning of W, does not degrade as significantly as the condition number, due to the higher norm of W.

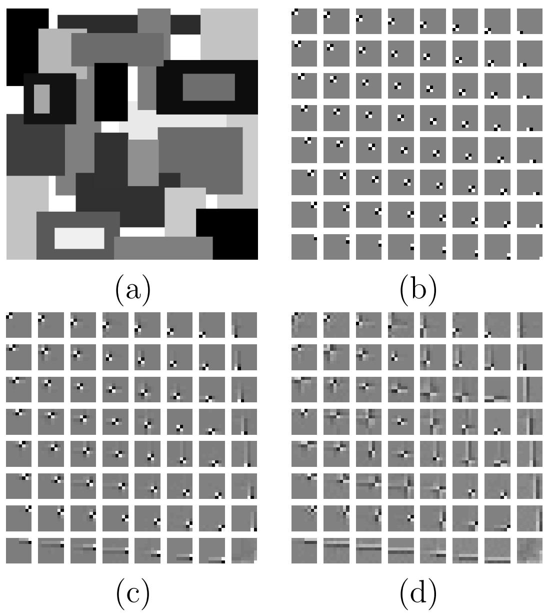


Figure 2.8: Piecewise-constant Case: (a) The Piecewise-constant image, (b) 2D Finite difference transform with $\kappa = 113$, (c) Learned transform with $\kappa = 15.35$, (d) Learned transform with $\kappa = 5.77$.

various ambiguities and for the success of sparsifying transform learning in applications.

Note that while the experiment for the $\mu = 0$ case was performed with zero-mean patches, we also observed inferior performance with $\mu = 0$ (compared to $\mu = \lambda$), when the means of the patches were retained.

2.4.4 Performance for Piecewise-Constant Images

Next, we consider learning a transform for a 512×512 piecewise-constant image (Figure 2.8(a)). Piecewise-constant images are well-sparsified by the finite difference transform. We work with 8×8 non-overlapping image patches in this experiment. The two-dimensional finite difference transform shown in Figure 2.8(b) is obtained as a Kronecker product of two one-dimensional finite difference matrices. (Each made square and non-singular by appending a row that has all 0's and a 1 on the last entry.) Note that this finite difference transform is square rather than overcomplete. It is an exact sparsifier for the patches (with means removed) of the image in Figure 2.8(a) for sparsity levels of $s \geq 5$. However, this transform is poorly conditioned with

$\kappa = 113$. We investigate the behavior of our algorithm for this interesting example. We solve Problem (P2.3) to learn transforms at various values of the parameter λ (with $\mu = \lambda$), with $s = 5$ and all other parameters fixed as in the experiment of Figure 2.4. We initialize the algorithm with the 2D finite difference transform itself, to check whether the algorithm converges to the same structure.

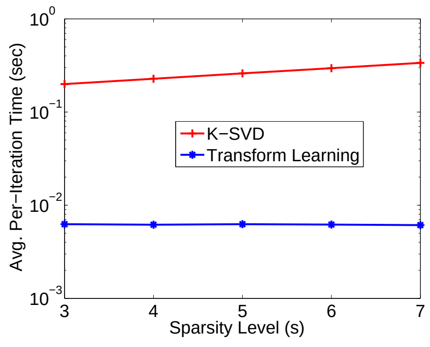
The learned transforms at $\lambda = 4 \times 10^2$ (Figure 2.8(c)) and $\lambda = 8 \times 10^3$ (Figure 2.8(d)) are well-conditioned with condition numbers 15.35 and 5.77, respectively. Both these transforms provide almost zero normalized sparsification errors for the data (normalized sparsification errors of 1.6×10^{-5} and 5×10^{-4} when κ is 15.35 and 5.77, respectively). Thus, our transform learning algorithm is able to learn well-conditioned transforms that sparsify almost as well as the poorly conditioned finite difference transform. Such well-conditioned adaptive transforms also perform better than poorly conditioned ones in applications such as image denoising [87, 2]. Note that the learned transforms do appear somewhat different from the finite difference transform, since they are adapted to the specific data.

When the transform learning algorithm was executed for the same data, but with lower sparsity levels $s < 5$, the learned transforms, apart from being well-conditioned, also provided significantly better sparsification at the same sparsity level, than the finite difference transform. The results here, much like previous ones, indicate the promise of our algorithm for (P2.3) to adapt to data and obtain significantly better sparse representations than analytical transforms.

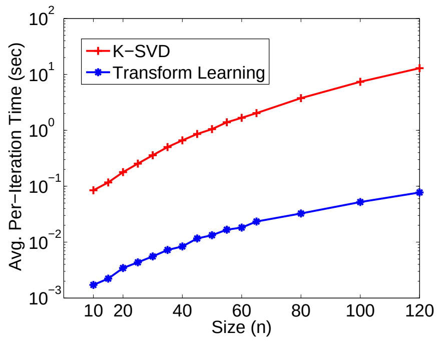
2.4.5 Run Time

Next, we test the execution times of the iterations of our algorithm (for Problem (P2.3)) and compare them with the execution times of the iterations of the popular synthesis dictionary learning algorithm, K-SVD [17, 86]. The goal here is to demonstrate the promising speed-ups of transform learning, which can prove advantageous in applications.

First, we study the behavior of run times as a function of the sparsity level s. At each sparsity level, we generate synthetic data similarly to the data of Figure 2.2, and execute the algorithm of Problem (P2.3). The algorithm



(a)



(b)

Figure 2.9: Algorithm Execution Time: (a) Average per-iteration execution time vs. sparsity level (s) for the synthesis K-SVD and our transform learning scheme, (b) Average per-iteration execution time vs. dictionary/transform size (n) for both the synthesis K-SVD and our transform learning scheme.

parameters are set similarly to the experiment of Figure 2.2 (note that $\lambda = 50$ for most of the experiments, and it is optimally set for the others). We also execute the synthesis K-SVD algorithm [17] with square dictionaries for the same data. The algorithms are executed for a fixed number of (700) iterations, and moreover the simulation is repeated five times (with fresh data) at each sparsity level.

Figure 2.9(a) plots the average per-iteration execution times of our algorithm and of the synthesis K-SVD as a function of the sparsity level. It can be seen that the iterations of our algorithm are significantly faster (by at least a factor of 32) than K-SVD. The per-iteration execution time of K-SVD increases quickly with sparsity level. In contrast, the per-iteration execution times of our algorithm are approximately the same at the various sparsity levels. The iterations of our algorithm are thus nearly 50-60 times faster than the synthesis K-SVD iterations at higher sparsity levels.

We also tested the speed of our algorithm as a function of transform size. The data for this experiment were generated synthetically at varying problem sizes (n). The number of training signals $N = 10n$, and the sparsity at which data are generated is $s = n/5$ (rounded to nearest integer). Both the synthesis K-SVD and our algorithm (for Problem (P2.3)) are executed for 100 iterations at the different problem sizes (both transform and dictionary are square). The conjugate gradient method within our algorithm is executed for 30 iterations, and the parameters such as μ, λ are set appropriately.

Figure 2.9 (b) plots the average per-iteration execution times of the algorithms as a function of problem size. The per-iteration execution times for our algorithm are at least about 50 times better than those for K-SVD. Moreover, the gap widens as the problem size increases. At $n = 120$, the iterations of the transform learning algorithm are 167 times faster than those of K-SVD. This was also predicted by the expressions for computational cost in Section 2.2. The K-SVD iteration time also increases at a much faster rate with problem size compared to our algorithm. The results indicate that the proposed transform learning can be easily implemented at large problem sizes.

Since our algorithm converges very quickly, we can say that sparsifying transforms can in general also be learned (using Problem (P2.3)) much faster than synthesis dictionaries (using K-SVD). Note that while we compared to the case of a square synthesis dictionary here, the speed-ups are much greater when compared to the overcomplete K-SVD.

We expect the run times of our algorithm to decrease substantially with conversion of the code to C/C++, and code optimization. Efficient implementation of sparse matrix multiplications (in the transform update step (2.12)) and efficient thresholding (i.e., thresholding training vectors in parallel and without resorting to full sorting in the sparse coding step) are just some of the ways to reduce run times.

2.4.6 Preliminary Denoising Experiments

Here, we test the effectiveness of our denoising algorithm that solves Problem (P2.4). For comparison, we also consider a denoising formulation involving synthesis dictionary learning as follows [1]:

$$\text{(P2.5)} \quad \min_{D,X,\hat{Y}} \left\|\hat{Y} - DX\right\|_F^2 + \rho \left\|Y - \hat{Y}\right\|_F^2 \quad \text{s.t.} \;\; \|X_i\|_0 \leq s \;\; \forall\, i \tag{2.15}$$

The parameter ρ is chosen similarly (i.e., inversely proportional to σ) to τ of Problem (P2.4). The synthesis denoising formulation (P2.5) is also solved by alternating minimization [1]. In one step, D and X are learned via K-SVD with fixed $\hat{Y}$ while in the other step, $\hat{Y}$ is updated by a least squares procedure.

The data for the comparison of problems (P2.4) and (P2.5) were generated synthetically using a random 20×20 synthesis dictionary ($n = 20$) with a sparsity level of 5 ($s = 5$). Note that the synthetic data obey both the synthesis dictionary model and the transform model exactly. The generated synthetic data were further corrupted by additive i.i.d. Gaussian noise. The number of noisy signals is $N = 10n$. The parameters τ and ρ for (P2.4) and (P2.5) were both optimally chosen (empirically) as $\frac{0.5}{\sigma}$. The parameter $\lambda = 20$ for our transform-based algorithm, and all other parameters for our algorithm (such as μ, etc.) were the same as for Figure 2.2.

Problems (P2.4) and (P2.5) were solved at various noise levels with fresh data generated at each noise level. In the case of Problem (P2.5), we learned both a square 20×20 dictionary, and an overcomplete 20×80 dictionary. At each noise level, the denoised signal-to-noise ratio (SNR), i.e., $20 \log_{10}\left(\|\hat{Y}\|_F / \|Y^* - \hat{Y}\|_F\right)$ is computed for both algorithms. Here, Y^* denotes the original noiseless data to which noise was added, and $\hat{Y}$ is the output of the denoising algorithm (using either (P2.4) or (P2.5)). The denoised SNR is averaged over five trials at each noise level.

Figure 2.10 plots the average denoised SNR (in decibels) as a function of the noise standard deviation (σ). At a low noise level (or noise standard deviation) of 0.063, our algorithm provides nearly 4.2 dB better denoised SNR compared to the square K-SVD, and 2.1 dB better denoised SNR compared to the four-fold overcomplete dictionary. (As expected, the overcomplete K-SVD performs better than the square K-SVD.) The improvement drops at higher noise levels as both algorithms degrade in performance. However, even at high noise levels ($\sigma = 0.3$), our algorithm still provides 1.1 dB

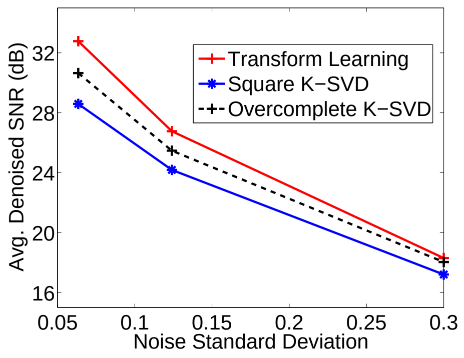


Figure 2.10: Signal denoising: Average denoised SNR (in dB) vs. noise standard deviation (σ) for our algorithm, and for K-SVD with square and overcomplete dictionaries.

of improvement over the square K-SVD, and 0.3 dB improvement over the overcomplete K-SVD, respectively. At mid-noise levels such as $\sigma = 0.124$, we obtain an improvement of 2.6 dB over the square K-SVD, and 1.3 dB over the overcomplete K-SVD, respectively. These results indicate that learned sparsifying transforms can provide promising denoising.

We have also applied sparsifying transforms to image denoising [2, 87] showing better denoising compared to both adaptive overcomplete synthesis and analysis dictionaries.

2.5 Conclusions

In this chapter, we studied a novel problem formulation for learning unstructured square sparsifying transforms from training data. The proposed alternating transform learning algorithm involves two steps - thresholding and nonlinear conjugate gradients. Our framework gives rise to well-conditioned transforms with much lower sparsification errors than analytical transforms. Results with natural images demonstrate that well-conditioning (but not necessarily unit conditioning) of the transforms is compatible with good sparsification and good performance in applications. Even for piecewise constant images, for which a difference operator provides optimal sparsification, but at high condition number, our well-conditioned learned transforms provide essentially identical, or even better sparsification. Our algorithm provides monotonic convergence of the cost function, and is insensitive to initialization. Moreover, the per-iteration computational cost of the proposed transform learning scheme is nearly two orders of magnitude lower than that of synthesis dictionary learning algorithms such as K-SVD. We have also introduced a signal denoising formulation involving sparsifying transform learning in this chapter, and demonstrated promising performance for the proposed algorithm.

CHAPTER 3

LEARNING DOUBLY SPARSE TRANSFORMS FOR IMAGES

While Chapter 2 discussed the learning and promise of unstructured square transforms, we will now turn to the study of a very interesting structured transform called the doubly sparse transform [1].

3.1 Doubly Sparse Transform Model

Imposing additional structure on the learned transform leads to computational advantages over the 'unstructured' transforms introduced in Chapter 2. We propose to learn a square ($n \times n$) sparsifying transform W that is a product of two different transforms (matrices), i.e., $W = B\Phi$, where $\Phi \in \mathbb{R}^{n\times n}$ is an analytical transform with an efficient implementation, and $B \in \mathbb{R}^{n\times n}$ is a transform that is constrained to be sparse (i.e., has few significant non-zero elements). Such a transform W is said to be 'doubly sparse', since it provides a sparse representation for data and has matrix B that is sparse.

The proposed doubly sparse transform structure combines the advantages of trained and analytic transforms: adapting to the data, it provides better representations and denoising than analytical transforms, yet owing to the sparsity of B, it can be stored and applied efficiently. As we show, it can also be learned more efficiently than unstructured transforms [9].

The structure $W = B\Phi$ is expected to be an effective sparsifying transform because Φ matrices such as the DCT when applied to natural images (or image patches) produce a result that is already approximately sparse. Therefore, by further modifying the result using only the limited number of degrees of freedom in a sparse transform B, one may be able to produce a highly sparse result.

[1]Parts of the material in this chapter have been previously published in [88] and [2].

Learning a *synthesis* dictionary composed of a product of an analytical dictionary and a sparse matrix has been proposed with a different motivation by Rubinstein et al. [60]. Their formulation is still non-convex and NP-hard, and their learning algorithm is a variation of K-SVD and thus suffers from similar drawbacks as K-SVD (i.e., lack of any convergence guarantees). Another recent work [89] learns synthesis dictionaries for the different bands in the wavelet domain of images. However, the atoms of those dictionaries are not constrained to be sparse, and it is unclear if the method can outperform previous work [60, 1, 56, 59] in applications.

3.1.1 Main Highlights

In this work, we focus on square transforms. We propose novel problem formulations for learning doubly sparse transforms that are well-conditioned. The main results of our work are enumerated as follows.

(i) When the sparsity of B is measured using a convex differentiable penalty, we guarantee the convergence of the objective in our proposed alternating algorithm. When the non-convex ℓ_0 "norm" is used to measure sparsity of B, we show the convergence of the objective and iterates in our algorithm empirically. These desirable convergence properties contrast with those of popular synthesis or analysis learning algorithms such as K-SVD, for which no such results, theoretical or empirical, are available.

(ii) The adapted doubly sparse transform $B\Phi$ provides a better representation of images than the analytical transform Φ.

(iii) Doubly sparse transforms can be learned for natural images with highly sparse B matrices, with only a marginal loss in image representation quality compared to unstructured (or non-sparse) transforms. In fact, imposing the doubly sparse property leads to much faster convergence of learning, and faster computations with the sparse transform.

(iv) The learned doubly sparse transforms have reduced storage requirement and generalize better than the unstructured transforms.

(v) The proposed doubly sparse transform learning enables the use of "cheap" analytical transforms Φ such as the Hadamard transform, with essentially the same image representation quality as with better Φ, but at a substantially lower computational cost.

(vi) Doubly sparse learning is robust to the size of the training set, and requires far fewer training signals than unstructured transform learning.

(vii) We present a novel adaptive sparsifying-transform-based image denoising formulation and algorithm in this chapter. The denoising performance of the learned doubly sparse transforms is comparable to, or better than, overcomplete K-SVD. Most importantly, doubly sparse denoising is much faster. The doubly sparse transforms also denoise faster than the unstructured transforms of the previous Chapter 2, with little loss in denoising quality. In fact, at high noise, the doubly sparse transforms mitigate overfitting to noise, and denoise somewhat better than unstructured transforms, which have more degrees of freedom.

The rest of this chapter is organized as follows. Our proposed problem formulations for doubly sparse transform learning are described in Section 3.2. Section 3.3 details the proposed algorithms for learning doubly sparse transforms and discusses their relevant properties such as convergence and computational cost. In Section 3.4, we introduce the image denoising formulation and algorithm incorporating adaptive sparsifying transforms. Section 3.5 demonstrates the performance of our algorithms in image representation and image denoising. In Section 3.6, we conclude.

3.2 Problem Formulations

3.2.1 Unstructured Transform Learning

In Chapter 2, we proposed a novel problem formulation for unstructured square transform learning (i.e., Problem (P2.3)). In this chapter, we use a simple variant of that formulation as follows:

$$\begin{aligned}
\text{(P3.1)} \quad & \min_{W,X} \; \|WY - X\|_F^2 - \lambda \log\, |\det W| + \mu \, \|W\|_F^2 \\
& s.t. \;\; \|X_i\|_0 \leq s \;\; \forall \; i
\end{aligned}$$

where we have used the absolute value of the determinant above. In Chapter 2, we imposed the restriction $\det W > 0$, since we can always switch from a W with $\det W < 0$ to one with $\det W > 0$ trivially by swapping two rows of W. We will work with the setting $\lambda = \mu$ above in this chapter, which results, in the limit $\lambda \to \infty$, in the optimizing transform having a spectral norm of $1/\sqrt{2}$. This setting has been shown to work well in practice in Chapter 2.

The properties of the unstructured square transform learning formulation have been discussed in detail in Section 2.1 of Chapter 2 [2]. An interesting property that we did not indicate in the previous chapter relates the norms of the rows (or, equivalently the columns) of the transform to its condition number. Denoting the i^{th} row of W by w_i, we have that

$$\max_{i,\,j} \; \frac{\|w_i\|_2 - \|w_j\|_2}{\|w_j\|_2} \; \leq \; \kappa(W) - 1 \tag{3.1}$$

The bound follows from the fact that the largest and smallest singular values of W obey $\sigma_1 \geq \max_i \|w_i\|_2$ and $\sigma_n \leq \min_i \|w_i\|_2$, respectively. The bound is tight when $\kappa(W) = 1$ (in which case all rows of W have identical norms), and it indicates that the norms of the rows of W can be controlled by controlling the condition number. This eliminates the need to have separate constraints on row or column norms of W like in the synthesis and analysis cases, where such a constraint is needed to eliminate the scaling ambiguity.

As discussed Chapter 2, unstructured transform learning offers explicit advantages over synthesis, and analysis dictionary learning. Specifically, the sparse coding problem (i.e., Problem (P3.1) with fixed W) admits an easy and exact solution, and (P3.1) also does not include a highly non-convex function of the product of two unknown matrices.

[2]The various properties of formulation (P2.3) in Chapter 2 trivially hold/extend with the introduction of the absolute value for the determinant in (P3.1).

3.2.2 Doubly Sparse Transform Learning

The proposed doubly sparse structure $W = B\Phi$ combines adaptivity with the efficiency of the analytical Φ. For example, the cost of applying the transform Φ to an n-dimensional signal scales as $O(n \log n)$ for the Hadamard, DCT, Discrete Fourier Transform (DFT), etc. We now formulate the learning of such a doubly sparse transform by replacing W with $B\Phi$ in Problem formulation (P3.1) with $\lambda = \mu$.

$$\text{(P3.2)} \quad \min_{B,X} \|B\Phi Y - X\|_F^2 - \lambda \log |\det(B\Phi)| + \lambda \|B\Phi\|_F^2$$
$$s.t. \quad \|B\|_0 \leq r, \; \|X_i\|_0 \leq s \;\; \forall\, i$$

where $\|B\|_0 \triangleq \sum_{i,j} 1_{\{B_{ij} \neq 0\}}$, with B_{ij} the entry of B from row i and column j, and $1_{\{B_{ij} \neq 0\}}$ is the indicator function of $B_{ij} \neq 0$. The l_0 "norm" constraint on matrix B enforces sparsity of the entries of B. We assume r non-zeros in B. As the sparsity level $r \to n^2$, Problem (P3.2) tends to Problem (P3.1), but instead involves learning a transform W that is decomposable as $B\Phi$. For invertible Φ, the two problems are exactly equivalent in the limit $r = n^2$.

Now, since the determinant is multiplicative, we obtain that

$$\log |\det(B\Phi)| = \log |\det B| + \log |\det \Phi| = \log |\det B| + C$$

where the constant $C = \log |\det \Phi|$. Moreover, if matrix Φ is orthonormal (for example, Wavelets, or DCT), we get that $\|B\Phi\|_F^2 = \|B\|_F^2$.

With these simplifications, the problem formulation (P3.2) becomes

$$\text{(P3.2)} \quad \min_{B,X} \left\|B\hat{Y} - X\right\|_F^2 - \lambda \log |\det B| + \lambda \|B\|_F^2$$
$$s.t. \quad \|B\|_0 \leq r, \; \|X_i\|_0 \leq s \;\; \forall\, i$$

where $\hat{Y} = \Phi Y$. Thus, the doubly sparse problem formulation (P3.2) is similar to (P3.1), but with the additional, yet crucial sparsity constraint. Note that for non-orthonormal Φ, we would still have the $\|B\Phi\|_F^2$ penalty in the cost.

Now, the set of matrices with unit condition number also includes scaled identity matrices, which are the sparsest possible matrices[3]. Therefore, as

[3]Since no non-singular matrix B exists for $r < n$, such sparsity levels are inadmissible

$\lambda \to \infty$ in (P3.2), the condition number of the optimal/minimizing transform(s) tends to 1, just like in the unstructured case [9].

As an alternative to (P3.2), we consider the following formulation, which replaces the l_0 constraint on B with a convex penalty $h(B)$ in the cost with weighting ζ. This formulation will be shown to lead to an algorithm with a convergence guarantee.

$$\text{(P3.3)} \min_{B,X} \left\| B\hat{Y} - X \right\|_F^2 - \lambda \log |\det B| + \lambda \|B\|_F^2 + \zeta h(B)$$
$$s.t. \;\; \|X_i\|_0 \leq s \;\; \forall\, i$$

For example, $h(B)$ can be the l_1 penalty $\|B\|_1 = \sum_{i,j} |B_{ij}|$. Alternatively, we consider a slightly smoothed l_1 penalty $h(B) = \sum_{i,j} \sqrt{B_{ij}^2 + \epsilon}$, where $\epsilon > 0$ is chosen sufficiently small. Such a penalty is amenable to optimization schemes that exploit the gradient. The relaxed formulation (P3.3) was empirically observed to perform similarly to Problem (P3.2). However, Problem (P3.2) with the exact sparsity constraint on B, while not enjoying the same convergence guarantee as (P3.3), enables faster learning (shown in Section 3.3).

Both Problems (P3.2) and (P3.3) admit an equivalence class of solutions [9]. Given a minimizer $(\hat{B}, \hat{X})$ with $\hat{X}$ sparse, we can form trivially equivalent minimizers by simultaneously permuting the rows of $\hat{B}$ and $\hat{X}$, or by premultiplying them with a diagonal ± 1 matrix.

One could constrain B in (P3.3) to be positive definite, i.e., $B \succ 0$. If in addition the l_0 "norm" for X were relaxed to an l_1 norm constraint or l_1 penalty, the resulting problem would be jointly convex in B and X. However, upon further investigation, we found that enforcing B to be positive definite is too restrictive, and provides almost no improvement over the analytical Φ. In fact, experimental results (see for example, Figure 3.2 and Section 3.5) show that B learned via (P3.2) has an approximately skew-symmetric off-diagonal, thus implying $B \not\succ 0$. We could rely on the observed properties of the learned B for natural images to design reasonable convex learning formulations. However, a discussion of such formulations is beyond the scope of this work and will be presented elsewhere.

for (P3.2).

3.2.3 Alternative Doubly Sparse Formulations

We have also studied the learning of an alternative doubly sparse structure $W = \Phi B$, which was, however, empirically observed to perform worse than the proposed $B\Phi$ structure in applications. Moreover, the structure $B\Phi$ has computational advantages over the alternative ΦB. For the former, the product $\hat{Y} = \Phi Y$ can be pre-computed once, before an optimization algorithm begins, whereas the product $\Phi(BY)$ would have to be computed repeatedly in iterative algorithms, when employing the latter ΦB structure.

Finally, while many more useful properties can be enforced on sparsifying transforms, these are beyond the scope of the current work.

3.3 Algorithms and Properties

3.3.1 Algorithms

Here, we outline algorithms for solving the doubly sparse transform learning problems (P3.2) and (P3.3). In earlier work on unstructured square transforms [9], we proposed an alternating algorithm for solving Problem (P3.1). Our algorithms for solving Problem (P3.2) or (P3.3) also alternate between updating X and B.

3.3.1.1 Sparse Coding Step

In this step, we solve Problem (P3.2) (or (P3.3)) with fixed B.

$$\min_{X} \left\| B\hat{Y} - X \right\|_F^2 \quad s.t. \quad \|X_i\|_0 \leq s \;\; \forall \; i \tag{3.2}$$

The solution X can be computed exactly by zeroing out all but the s coefficients of largest magnitude in each column of $B\hat{Y}$. We refer to this operation as *projection onto the ℓ_0 ball.*

3.3.1.2 Transform Update Step

In this step, we solve Problem (P3.2), or (P3.3) with fixed X. For Problem (P3.3), the transform update step involves the following problem, with

$h(B) = \sum_{i,j} \sqrt{B_{ij}^2 + \epsilon}$.

$$\min_{B} \left\| B\hat{Y} - X \right\|_F^2 - \lambda \, \log \, |\det B| + \lambda \, \|B\|_F^2 + \zeta h(B) \tag{3.3}$$

This unconstrained minimization problem is non-convex but smooth. One could solve it using iterative procedures such as gradient descent, or the conjugate gradient (CG) algorithm [83] (which typically converges faster). The CG method can be employed with backtracking line search [83] (Armijo step size rule), which guarantees decreasing cost (and thus prevents the algorithm from entering the barrier region). Fixed (sufficiently small) step size rules were also empirically observed to work well and faster. The CG method converges quickly, and can be executed for a fixed number of iterations in practice, to save run time.

In the case of Problem (P3.2), the transform update step solves the following optimization problem.

$$\begin{aligned} &\min_{B} \left\| B\hat{Y} - X \right\|_F^2 - \lambda \, \log \, |\det B| + \lambda \, \|B\|_F^2 \\ &s.t. \;\; \|B\|_0 \leq r \end{aligned} \tag{3.4}$$

This constrained problem does not have have an analytical solution. Moreover, the constraint set is non-convex. One could perform the transform update using the iterative projected gradient algorithm, or projected CG [83]. However, we found that the alternative heuristic strategy of employing a few iterations of the standard CG algorithm followed by post-thresholding led to better empirical performance in terms of the metrics defined in Section 3.5.1. Hence, we choose this alternative strategy. Matrix B is essentially thresholded after some iterations of CG, and we retain only the r elements of largest magnitude.

The gradient expressions for the various terms in the objective function of (P3.2) or (P3.3) (used for CG) are listed below for completeness.

$$\nabla_B \log \, |\det B| = B^{-T} \tag{3.5}$$

$$\nabla_B \|B\|_F^2 = 2B \tag{3.6}$$

$$\nabla_B \|BY - X\|_F^2 = 2BYY^T - 2XY^T \tag{3.7}$$

$$\nabla_{B_{ij}} \left(\sqrt{B_{ij}^2 + \epsilon} \right) = \frac{B_{ij}}{\sqrt{B_{ij}^2 + \epsilon}} \tag{3.8}$$

When the transform update step of (P3.2) is solved using the proposed heuristic strategy, then the determinant of B needs to be monitored after the post-thresholding, to ensure that the algorithm does not enter the log barrier. If the thresholding produces a B with $\det(B) = 0$, we perturb B away from $\det(B) = 0$ by adding a random matrix of small l_2 norm. In all our experiments (Section 3.5), we observed that with reasonable initializations for B, such as the identity matrix, the aforementioned extra step was never invoked by the algorithm for (P3.2), as it did not reach the degenerate state.

3.3.2 Convergence

Here, we discuss the convergence of the proposed doubly sparse transform learning algorithms. The cost functions in our formulations are all lower bounded [9].

The algorithms alternate between sparse coding and transform update steps. The solution for the sparse coding step of (P3.2) and (P3.3) is exact. Therefore, the respective cost functions decrease in this step.

For the transform update step of (P3.3), the solution is obtained by conjugate gradients (for instance with Armijo step size rule). Thus, in this step, the cost function can again only decrease for (P3.3). The cost function being monotone decreasing and lower bounded, it must converge for the alternating algorithm for (P3.3).

On the other hand for (P3.2), the transform update step involving CG followed by post-thresholding is a heuristic approach. Hence, we cannot guarantee convergence of the cost in this case. However, as illustrated empirically in Section 3.5, both the cost and the iterate converge for the alternating algorithm for (P3.2). It will be also demonstrated empirically in Section 3.5 that for reasonably sparse B (small r), the algorithm for Problem (P3.2) converges much faster (i.e., in fewer iterations) than that for (P3.1). We hypothesize that this is because, for small r, doubly sparse transforms have far fewer free parameters than unstructured transforms.

3.3.3 Computational Cost

We now demonstrate the low computational cost of the proposed algorithms. We estimate the cost of each step of our algorithms. The sparse coding step in Problem (P3.2) requires rN multiply-add operations to compute $B\hat{Y}$, when B has r non-zeros. Since $r = \beta n^2$, where typically the non-zero fraction $\beta \ll 1$, the preceding cost becomes $\beta n^2 N$. The projection of $B\hat{Y}$ onto the ℓ_0 ball (3.2), if done by full sorting [85] would involve $O(nN \log n)$ computations. Thus, the sparse coding step of Problem (P3.2) has a cost of $Nn(\beta n + C_2 \log n)$, where C_2 is a constant. For $\beta < 1$, this is lower (better) than the cost of the sparse coding step for the 'unstructured' Problem (P3.1) [9]. Thus, the doubly sparse problem formulation can provide speed-ups over the unstructured (P3.1) in learning. In any case, the cost of sparse coding is dominated by $\beta n^2 N$.

When the doubly sparse learning involves the convex sparsity penalty for B (Problem (P3.3)), the matrix B in each iteration may not be exactly sparse (may only be compressible). Hence, the computational speed-ups in the sparse coding step for this case are typically lower than with the l_0 "norm".

Other than the extra post-thresholding in (P3.2), the algorithms for both (P3.1) and (P3.2) use conjugate gradients in the transform update step. Hence, since the objective functions of (P3.2) and (P3.1) are the same, so are the costs of the respective conjugate gradient steps, which are roughly $\alpha N n^2 + (1+C_3)Jn^3$ [9]. Here, $\alpha = s/n$, J is the number of conjugate gradient steps, and C_3 is a constant. The $\alpha N n^2$ cost arises from the computation of XY^T (3.7) (computed once at the beginning of the transform update step), while the n^3 costs arise from computing B^{-T} (3.5) and BYY^T (YY^T assumed pre-computed once at a total cost of Nn^2 for the entire algorithm). The post-thresholding of B in the transform update step of (P3.2) if done by sorting would require $O(n^2 \log n)$ operations. Since $\log n \ll N$ typically, and assuming that $(1+C_3)Jn < \alpha N$, the cost per transform update step of (P3.2) is dominated by $\alpha N n^2$. The cost per transform update step of (P3.3) scales similarly. (Monitoring the determinant of B after the post-thresholding step of (P3.2) would take $O(n^3)$ operations, but this is typically unnecessary for reasonable initializations for B.)

The total cost per iteration (of sparse coding and transform update) of the proposed doubly sparse transform learning algorithms is thus roughly

$(\alpha + \beta)Nn^2$. (To account for the non-exact sparsity of B in (P3.3), the non-zero fraction β needs to be increased in this expression in the case of (P3.3).) In contrast, for the unstructured transform learning case [9], the cost per iteration is roughly $(\alpha+1)Nn^2$. A direct comparison of the costs indicates the computational efficacies of the proposed doubly sparse transform learning. The smaller the β, the greater the speedup that can be expected for doubly sparse learning. Nonetheless, the cost per algorithm iteration scales in order as $O(n^2N)$ for all the transform learning algorithms.

Compared to the computational cost of synthesis or analysis learning, the costs of the proposed doubly sparse transform learning algorithms are significantly lower. We have shown [9] that the computational cost per-iteration of our algorithm for (P3.1) is much lower than the per-iteration cost of synthesis dictionary learning algorithms such as K-SVD (which scales as $O(n^3N)$ for square dictionaries). A similar advantage exists over the per-iteration cost of analysis K-SVD [26]. The actual per-iteration run times for (P3.1) were also shown [9] to be orders of magnitude smaller than for synthesis K-SVD. Since, as argued above, the algorithm for Problem (P3.2) is cheaper than for (P3.1), we can expect even shorter per-iteration run times for doubly sparse learning compared to K-SVD.

3.4 Application to Image Denoising

3.4.1 Problem Formulation

Image denoising is a widely studied image processing problem of estimating an image $x \in \mathbb{R}^P$ (2D image represented as a vector) from its measurement $y = x+h$ corrupted by noise h. We focus in this work on adaptive sparsifying-transform-based image denoising. Our goal here is an initial exploration of the potential of the proposed framework and algorithms for learning a sparsifying transform in this classical and prototypical application.

Similar to previous dictionary-based denoising approaches [1, 59, 25, 70], we work with overlapping image patches, and learn a transform adapted to them. We propose a simple problem formulation (similar to the one proposed for signal denoising [9]) for denoising image patches using sparsifying transforms, as follows.

$$\min_{W,\{x_i\},\{\alpha_i\}} \sum_{i=1}^{M} \|Wx_i - \alpha_i\|_2^2 + \lambda Q(W) + \tau \sum_{i=1}^{M} \|R_i\, y - x_i\|_2^2$$
$$s.t. \;\; \|\alpha_i\|_0 \leq s_i \;\; \forall\, i \qquad \text{(P3.4)}$$

Here, $Q(W)$ represents the portion of the cost depending on only W. For the doubly sparse case, $Q(W) = Q(B)$, since $W = B\Phi$. For example, the $Q(W)$ corresponding to (P3.2) includes the log-determinant and Frobenius norm penalties, along with the indicator function of the set of r-sparse matrices. Vector $R_i y$ in (P3.4) denotes the i^{th} patch of image y (M overlapping patches are assumed), with $R_i \in \mathbb{R}^{n\times P}$ being the operator that extracts it as a vector from the image. We assume that the noisy patch $R_i\, y$ can be approximated by a noiseless version x_i that is approximately sparsifiable by an adaptive transform (the *noisy signal transform model* [9], see also [90]). Vector $\alpha_i \in \mathbb{R}^n$ denotes the transform sparse code of x_i with an a priori unknown number s_i of non-zeros.

The denoising model here is different from the model assumed in Problem (P3.2), since the latter is aimed at sparse image representation (without requiring explicit denoising). The parameter τ in (P3.4) is typically inversely proportional to the noise level σ [1]. When $\sigma = 0$, the optimal $x_i = R_i\, y$, and (P3.4) reduces to a transform learning problem. Once the denoised patches x_i are found, the denoised image x is obtained by averaging the x_i's at their respective locations in the image (cf. [5] for a similar technique).

3.4.2 Algorithm

Problem formulation (P3.4) denoises the patches using an adaptive W, and is non-convex. The algorithm that we propose to solve Problem (P3.4) iterates over a transform learning step and a variable sparsity update step. Once the iterations complete, there is a denoised image update step.

3.4.2.1 Transform Learning

In this step, we fix $x_i = R_i y$ and $s_i = s$ (fixed s initially) for all i in (P3.4), and solve for W and α_i ($i = 1, 2, ..., M$) using our proposed transform learning algorithms.

3.4.2.2 Variable Sparsity Update

In this step, we update the sparsity levels s_i for all i.

For fixed W and α_i $(i = 1, 2, ..., M)$, (P3.4) reduces to a least squares problem in the x_i's. Each x_i can then be independently updated as follows, where † denotes the pseudo-inverse.

$$x_i = G \begin{bmatrix} \sqrt{\tau} R_i y \\ \alpha_i \end{bmatrix}, \text{where } G = \begin{bmatrix} \sqrt{\tau} I \\ W \end{bmatrix}^{\dagger} \tag{3.9}$$

However, we do not fix α_i in (3.9), and instead only let it be a thresholded version of $W R_i\, y$, and determine the sparsity level s_i. Let $H_{s_i}(b)$ denote the operator that zeroes out all but the s_i elements of largest magnitude of $b \in \mathbb{R}^n$. Then, our assumption is that $\alpha_i = H_{s_i}(W R_i y)$ for some s_i that is unknown a priori.

We choose the sparsity s_i for the i^{th} patch such that the error term $\|R_i\, y - x_i\|_2^2$ computed after updating x_i by (3.9) (with α_i held at $H_{s_i}(W R_i y)$) is below $nC^2\sigma^2$ [1] (the error term decreases to zero, as $s_i \nearrow n$), where C is a fixed parameter. This requires repeating the least squares update (3.9) of x_i for each i at various sparsity levels incrementally, to determine the level at which the error term falls below the required threshold.

We propose an efficient way to do this: add one non-zero element at a time from $W R_i\, y$ (elements chosen in descending magnitude ordering) to α_i in (3.9), until the error measure $\|R_i\, y - x_i\|_2^2$ with the newly updated x_i falls below the required threshold. The matrix G in (3.9) can be pre-computed, and the addition of a new non-zero element to α_i leads to x_i being updated by adding a column of G scaled by the new non-zero element.

Once the variable sparsity levels s_i are chosen for all i, we use the new s_i's back in the transform learning step, and iterate over the learning and variable sparsity update steps, which leads to a better denoising performance compared to one iteration. In the final iteration, the x_i's that are computed (satisfying the $\|R_i\, y - x_i\|_2^2 \leq nC^2\sigma^2$ condition) represent the denoised patches.

```
Image Denoising Algorithm
Input :   $y$ - Noisy Image, $s$ - fixed sparsity, $L$ - number of iterations
Output :   $x$ - Denoised image
Initialization :   Patches $x_i = R_i y$, $s_i = s$ for $i = 1, 2, ...., M$
For  k = 1:L Repeat
    1. Learn $W$ and $\alpha_i$ for patches $x_i = R_i y$ with known sparsities $s_i$, for
       all $i = 1, 2, ...., M$.
    2. Update $s_i$ for all $i = 1, 2, ...., M$ :  Increase $s_i$ in $\alpha_i = H_{s_i}(W R_i y)$ in
       (3.9), until the error condition $\|R_i\, y - x_i\|_2^2 \leq nC^2\sigma^2$ is reached.
End

Update $x$ : Average the denoised patches $x_i$ at respective image locations.
```

Figure 3.1: Algorithm to denoise images using (P3.4).

3.4.2.3 Denoised Image Update

Once the denoised patches x_i are found using the proposed scheme, they are restricted to their range (e.g., 0-255), and averaged at their respective image locations to produce the denoised image x. The image denoising algorithm is summarized in Figure 3.1.

We work with mean subtracted patches during optimization, and the means are added back to the final denoised patches. We also typically learn on a subset of all patches selected uniformly at random [59]. In this case, the update of s_i's is only performed on a subset of patches, except in the final denoising iteration when all the s_i's are updated.

In Problem (P3.4), we denoise the image patches without directly enforcing the constraint $R_i x = x_i\ \forall\ i$. In the end, we obtain the least-squares solution for x in the set of equations $R_i x = x_i\ \forall\ i$ (with the x_i's here denoting the denoised patches) . This results in the averaging of patches that produces x. This technique, although sub-optimal, results in a highly efficient algorithm. As will be shown in Section 3.5, the proposed algorithm indeed provides promising denoising performance at a low computational cost.

An interesting observation is that the data fidelity term $\sum_{i=1}^{M} \|R_i\, y - x_i\|_2^2$ in the objective of (P3.4) becomes a scaled log-likelihood with the explicit constraint [4] $R_i x = x_i\ \forall\ i$, and assuming that all overlapping patches (i.e.,

[4]As mentioned earlier, although we do not enforce the constraint $R_i x = x_i\ \forall\ i$ directly in (P3.4), we do obtain the denoised x in the end as the least-squares solution in the set of equations $R_i x = x_i\ \forall\ i$ (with the x_i's here the denoised patches).

periodically positioned 2D image patches, with 1 pixel shifts between them) including ‘wrap around’ patches [5] (i.e., patches that begin near the right or bottom edges of an image are allowed to wrap around on the other side of the image (to complete them)) are included in our formulation. In that case, the data fidelity term is $\sum_{i=1}^{M} \|R_i\, y - x_i\|_2^2 = \sum_{i=1}^{M} \|R_i\, y - R_i x\|_2^2 = n\, \|y - x\|_2^2$. This is just a scaled version of the log-likelihood (the likelihood is denoted as $p(y|x)$) for the case when the noise (h) in the image pixels is i.i.d. Gaussian (in which case $p(y|x)$ is a Gaussian distribution or density function). In fact, in our experiments in Section 3.5, we will consider (denoising) images that have been corrupted with (simulated) i.i.d. Gaussian noise.

3.5 Numerical Experiments

3.5.1 Framework

We present results demonstrating the promise of our doubly sparse transform learning framework for image representation and image denoising. We first illustrate the convergence of the cost function and iterates for our alternating algorithms. Next, we demonstrate the effect of different choices of the analytical transform Φ. We then show that highly sparse transforms B can be learned for natural images with only a marginal loss in image representation quality as compared to unstructured transforms, and in fact, with dramatic gains in speed of learning. The doubly sparse transforms will be shown to be substantially better than analytical ones such as the DCT for image representation. The representation performance of adaptive transforms will be shown to improve with increasing patch size. Moreover, the doubly sparse transforms will be shown to require fewer training signals for learning than unstructured transforms. We also discuss the generalizability of learned doubly sparse transforms and their promise for image compression[5]. Finally, we demonstrate the promise of learned doubly sparse transforms in image denoising, where they perform comparably to, or better than, learned overcomplete synthesis dictionaries, while being much faster.

[5]The study of a complete compression scheme involving learned transforms is beyond the scope of this work. Instead, we use the quality of sparse image representation provided by the learned transforms as a surrogate to indicate their potential for compression.

We work with the l_0 "norm" for sparsity of X and B in the experiments (i.e., Problem (P3.2)), but this can be easily substituted by alternative sparsity penalties. We use a fixed step size in the transform update step of our algorithms. In our experiments, we initialize learning with the identity matrix for B, which we found to work best [6]. All implementations were coded in Matlab v7.8 (R2009a).

The data is generated as patches of natural images. We employ our doubly sparse transform learning Problem formulations for learning adaptive sparse representations of these patches. The means (or DC values) of the patches are removed and we only sparsify the mean-subtracted patches which are stacked as columns of Y (patches represented as vectors). The means are added back for image display. Mean removal is typically adopted in image processing applications such as K-SVD-based image denoising.

Performance Metrics: We introduce several metrics to evaluate the quality of learned transforms. We define the *normalized sparsification error* (NSE) as $\|B\hat{Y} - X\|_F^2 / \ \|B\hat{Y}\|_F^2$. This measures the fraction of energy lost in sparse fitting. In other words, it indicates the degree of energy compaction achieved in the transform domain, or how well the doubly sparse transform model holds for the signals. This is an interesting property to observe for the adaptive transforms. We also define the *normalized recovery error* as $\|Y - W^{-1}X\|_F^2 \ / \ \|Y\|_F^2$. This measures the normalized error in recovering the data Y as $W^{-1}X$ from their sparse codes X obtained by projecting $WY = B\Phi Y$ onto the ℓ_0 ball (3.2). This metric serves as a good surrogate for the performance of the learned transform in a compression application. A closely related metric is the recovery peak signal to noise ratio (or *recovery PSNR* (rPSNR)) defined (in dB) as the scaled (by the factor 20) base-10 logarithm of $255\sqrt{P} / \ \|Y - W^{-1}X\|_F$, where P is the number of image pixels. The *condition number* (CN) of $W = B\Phi$ and the *sparsity* of B are the other measures of usefulness and efficiency of the doubly sparse transform. An important parameter in our results is the transform sparsity fraction $\beta = \frac{r}{n^2}$, which we express in this section as a percentage.

[6]Initializing the sparse matrix B with identity is equivalent to initializing the doubly sparse matrix W with the analytical transform Φ, which already provides reasonable sparsification.

3.5.2 Convergence and Learning

For the first experiment, we extract the non-overlapping patches of size $\sqrt{n} \times \sqrt{n} = 8 \times 8$ ($n = 64$) from the 512×512 Barbara image [17]. The data matrix Y in this case has 4096 training signals (patches represented as vectors), and we work with $s = 11$. We fix the transform Φ in the experiment to be the 2D DCT (i.e., $\Phi = \Phi_0 \otimes \Phi_0$, where Φ_0 is the $\sqrt{n} \times \sqrt{n}$, or the 8×8 1D DCT matrix, and "$\otimes$" denotes the Kronecker product). Problem (P3.2) is solved to learn a sparse transform B that is adapted to the image patches. The algorithm parameters are $\lambda = 4 \times 10^5$, and $r = 0.25 \times n^2$ ($\beta = 25\%$). The conjugate gradient algorithm for transform update was run for 30 iterations with a fixed step size of 2×10^{-9}.

Figure 3.2 shows the progress of the algorithm over iterations. The objective function, sparsification error, and condition number are plotted over the iterations in Figures 3.2(a), 3.2(b), and 3.2(c) respectively. They all converge quickly. The condition number of B, which is also the condition number of $W = B\Phi$ for orthonormal Φ, converges to a low value of 1.41, indicating well-conditioning of the learned transform. The horizontal line in the sparsification error plot corresponds to the sparsification error of the patch-based 2D DCT (i.e., $W = \Phi = DCT$) at $s = 11$. Our algorithm improves/decreases the sparsification error by 5.8 dB compared to the analytical DCT.

The normalized sparsification error and normalized recovery error for the learned transform $W = B\Phi$ (with X obtained by projecting WY onto the ℓ_0 ball (3.2)) are 0.0450 (or 4.5%) and 0.0474, respectively. The two errors are similar for well-conditioned transforms. On the other hand, the normalized errors (both sparsification and recovery) corresponding to the 2D DCT for the zero mean Y are worse, at 0.0676. The results here indicate that image patches admit reasonably good doubly sparse transform representations.

The sparse approximation image recovered as $W^{-1}X$ using the learned W, has a recovery PSNR of 34.39 dB. In contrast, the image recovered using the fixed, patch-based 2D DCT (i.e., $W = \Phi = DCT$) has a worse PSNR of 32.85 dB.

The learned sparse matrix B has the structure of a positive diagonal matrix (Figure 3.2(e) shows the diagonal elements of B) with added off-diagonal perturbation. The off-diagonal perturbation represents the modification to the analytical Φ, and is approximately skew-symmetric (Figures 3.2(g) and

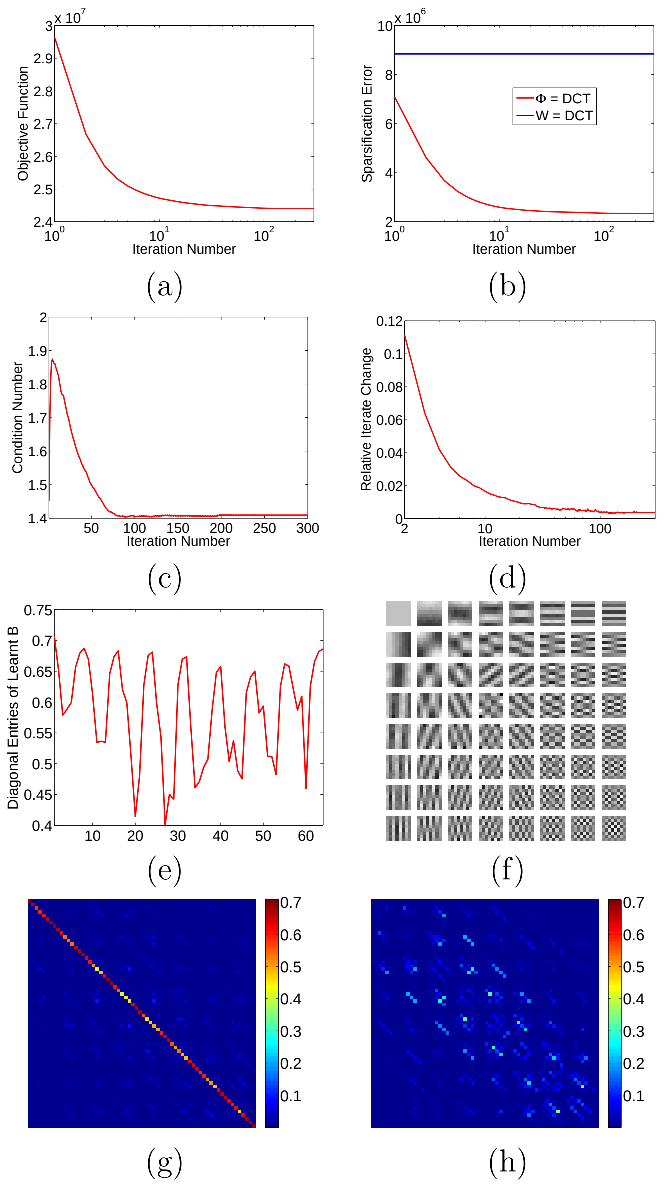


Figure 3.2: Behavior of the doubly Sparse Learning Algorithm for (P3.2): (a) Objective function, (b) Sparsification error when $\Phi = DCT$, along with the sparsification error (horizontal line) of the DCT by itself, (c) Condition number of B, (d) Relative iterate change, (e) Diagonal entries of the learned sparse B, (f) Rows of the learned transform $W = B\Phi$ shown as patches, (g) Magnitude of the symmetric part $(\frac{B+B^T}{2})$ of the learned B, (h) Magnitude of skew-symmetric part $(\frac{B-B^T}{2})$ of the learned B.

3.2(h) show most of the off-diagonal energy in the skew-symmetric part of the learned B).

Figure 3.2(f) shows the learned transform $W = B\Phi$, with each row shown as an 8×8 patch called the 'transform atom'. The learned doubly sparse transform exhibits geometric and frequency like structures, that sparsify Barbara.

Finally, we plot in Figure 3.2(d), the evolution of $\|B_i - B_{i-1}\|_F \,/\, \|B_{i-1}\|_F$, where i denotes the algorithm iteration number. This quantity measures the relative change between successive iterates/transforms. It quickly decreases to a low value indicating convergence of the iterates.

Stopping Condition and Initialization. The relative iterate change can be used as a stopping condition for the algorithm for (P3.2). Hence, in the following experiments, the algorithm terminates when the relative change of successive iterates (B) is less than a small threshold empirically chosen as 0.1%. We have observed that a smaller threshold increases the iteration count while providing only marginal improvement in the results.

Additionally, in the following experiments, the initial (40) iterations of the algorithm are executed with no post-thresholding of the transform in the transform update step. This usually results in better initialization, and thus convergence to better solutions.

3.5.3 Performance for Different Φ Matrices

We now test the performance of our algorithm for (P3.2) on the data of Figure 3.2 for different choices of the matrix Φ. All algorithm parameters are the same as for the experiment of Figure 3.2.

Table 3.1 shows the normalized sparsification errors (NSE-I), recovery PSNRs (rPSNR-I for Barbara), and condition numbers (CN-I) for the learned transforms (also referred to as *image-specific transforms* since they are adapted to a specific image) with 2D DCT, 2D Hadamard, 2D Haar Wavelets[7], and identity matrix as Φ matrices. The table also lists the performance metrics (abbreviated as NSE-F and rPSNR-F) for the various Φ

[7]Here, the 2D Haar transform matrix Φ is obtained as the Kronecker product of two 1D 3-level (since $\log_2(\sqrt{n}) = 3$) Haar matrices. Alternatively, one could perform the 3-level 2D Haar transformation as a sequence of three 1-level transformations, each obtained by using the Kronecker product of two 1D 1-level Haar matrices. In this case, the 1-level transformation in each step of the sequence is applied only on the approximation coefficients (in the first step, this is the entire image patch) from the previous step. However, this alternative construction of the Wavelet Φ led to inferior performance metrics, and is hence not included in Table 3.1.

Φ	CN-I	NSE-I	rPSNR-I	NSE-F	rPSNR-F
DCT	1.40	0.0456	34.37	0.0676	32.85
Hadamard	1.50	0.0467	34.24	0.1156	30.52
Wavelets	1.65	0.0574	33.48	0.1692	28.86
Identity	2.92	0.1193	31.12	0.5145	24.03

Table 3.1: Condition numbers (CN-I), normalized sparsification errors (NSE-I), and recovery PSNRs (rPSNR-I) for the (Barbara) image-specific transforms for various Φ matrices at $\beta = 25\%$, along with the normalized sparsification errors (NSE-F) and recovery PSNRs (rPSNR-F) for the fixed transforms Φ.

matrices themselves at the same sparsity level s as in the experiment of Figure 3.2.

All the learned doubly sparse transforms $W = B\Phi$ provide better normalized sparsification errors and recovery PSNRs than the corresponding analytical transforms Φ. The learned doubly sparse transforms with the 2D DCT and 2D Hadamard Φ matrices differ only slightly in their sparsification, recovery performance, although the 2D DCT matrix itself performs significantly better than Hadamard. Importantly, the operations involving the Hadamard matrix are faster since its entries are ± 1. Therefore, the doubly sparse learning allows us to exploit the inexpensive but poor Hadamard transform without loss of performance.

The results with Φ = Wavelets are quite inferior to both the DCT and Hadamard for the Barbara image. Furthermore, among the various learned doubly sparse transforms, the one learned with identity as Φ matrix performs the worst. Since this case essentially corresponds to a 'self-sparse' transform (i.e., the transform $W = B\Phi = B$ is sparse), the results indicate that doubly sparse transforms can perform significantly better than a self-sparse transform. On the other hand, the learned self-sparse transform is significantly better than the identity matrix, which by itself provides an unacceptable normalized error and recovery PSNR of 51% and 24 dB, respectively.

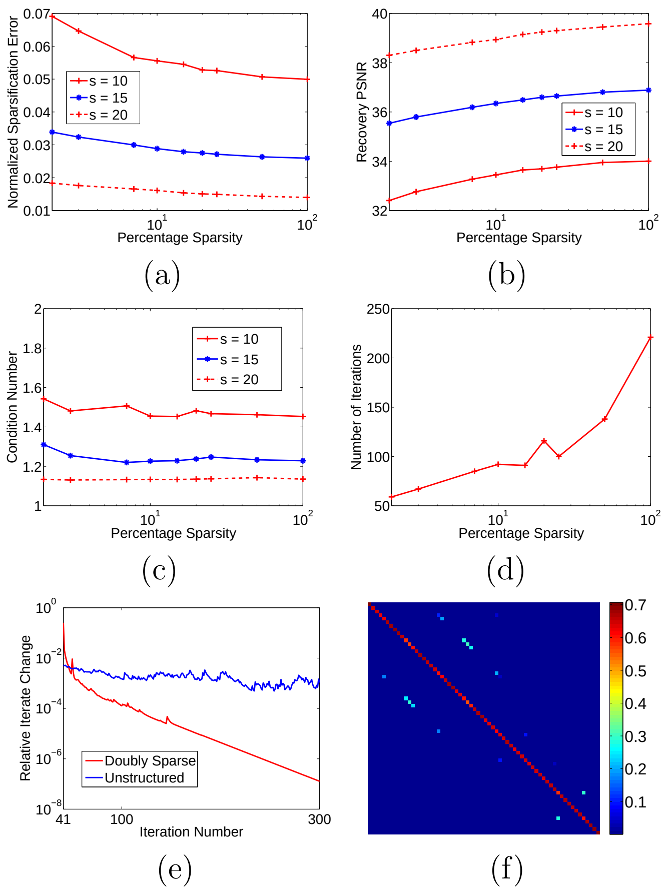


Figure 3.3: Behavior of algorithm as function of β for different values of s: (a) Normalized Sparsification error, (b) Recovery PSNR, (c) Condition number, (d) Algorithm iteration count for $s = 15$, (e) Relative iterate change vs. iteration count for $s = 15$ at transform sparsity fractions $\beta = 2\%$ and $\beta = 100\%$ (i.e., unstructured), (f) Magnitude of learned B at $\beta = 2\%$.

3.5.4 Algorithm Behavior as Function of Parameters

3.5.4.1 Performance as Function of s and β

We now work with the same data as for Figure 3.2, and test the performance of the algorithm for (P3.2) as a function of the data sparsity level s and transform sparsity fraction β. All other parameters are fixed as in the experiment of Figure 3.2 (except that a larger step size of 10^{-8} is used for large values of β, such as 50%, to ensure faster algorithm convergence).

Figure 3.3 plots the performance metrics of the learned transform as a function of the transform sparsity fraction β, at different data sparsity levels

	$s = 10$	$s = 15$	$s = 20$
NSE	0.0782	0.0393	0.0211
rPSNR	32.22	35.21	37.91

Table 3.2: Normalized sparsification errors and recovery PSNRs for the Barbara image with the patch-based 2D DCT.

$s = 10, 15, 20$. The condition number (Figure 3.3(c)) shows little change as a function of β. However, the normalized sparsification error (Figure 3.3(a)) increases monotonically and the recovery PSNR (Figure 3.3(b)) decreases as the transform sparsity fraction β is reduced. This behavior is expected because as its sparsity level decreases, the transform has fewer degrees of freedom to adapt (i.e., the learning is more constrained at lower transform sparsity levels). Likewise, all the metrics also degrade when the data sparsity level s is reduced. However, the values of all the metrics (for fixed s) are reasonable even at low transform sparsity levels such as β of 10-15 %, in the sense that the values at 10% are not too different from the corresponding values at 100%. This indicates that B can be constrained to be sparse with only a marginal loss in the performance metrics. A good choice of β would also depend on the transform Φ, since some Φ matrices sparsify the data much better than others.

For comparison, Table 3.2 lists the performance metrics obtained with the 2D DCT for the same data. For a particular sparsity level s, the learned doubly sparse transforms have better normalized sparsification errors and recovery PSNRs (as seen in Figure 3.3) than the 2D DCT even at very low transform sparsities such as 2%. This indicates the promise of efficient adaptive transforms over analytical ones.

Next, we turn to computational cost. We plot the number of algorithm iterations (Figure 3.3(d)) required to reach the stopping condition, as a function of β for $s = 15$. (The behavior is similar for other s values.) Depending on the value of β, the number of required algorithm iterations is reduced 2-4 fold compared to the unstructured transform case, i.e., 100% transform sparsity. (This reduction is not an exact monotone function of β, which is possibly an artifact due to the specific choice of the stopping criterion.) In general, the reduction is even greater with careful choice of parameters. In order to better illustrate the accelerated convergence of doubly sparse transform learning, we also plot the relative iterate change itself over the iterations

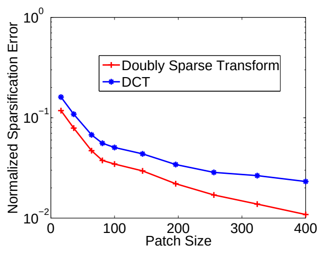
(Figure 3.3(e)) at $\beta = 100\%$, and another low $\beta = 2\%$. The convergence rate of the iterates is seen to be much better for the latter doubly sparse case. The learned sparse B at $\beta = 2\%$ is also shown (Figure 3.3(f) shows the magnitudes of elements of B). Thus, doubly sparse transforms, which have fewer free parameters, can in general be learned much faster than the unstructured transforms.

At smaller values of β, the per-iteration computational cost is also lower, since the operations involving sparse matrices are faster. Currently, the run time per iteration is up to 2 times lower at smaller values of β compared to $\beta = 100\%$. We expect the speedups to be much greater with more efficient implementation of sparse matrix operations, conversion of the code to C/C++, and code optimization.

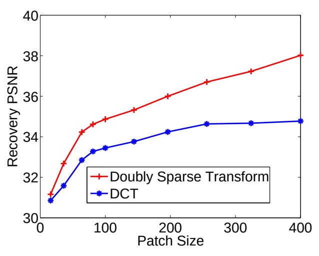
When $r = n$ ($\beta = 1/n$), we can only learn doubly sparse transforms W whose rows are scaled versions of the rows of the corresponding Φ (rows of Φ that sparsify the data better are likely to get larger scalings than those that sparsify worse), due to the log determinant penalty in (P3.2), which needs to be finite. Moreover, when the condition number of B is 1, such learned transforms are merely scalar multiples of Φ. Therefore, as r approaches n with the condition number of B kept close to 1, the normalized sparsification errors and recovery PSNRs of the learned doubly sparse transforms would approach the corresponding values for the analytical transform Φ. The results of Table 3.2 and Figure 3.3 corroborate this.

3.5.4.2 Performance as Function of λ

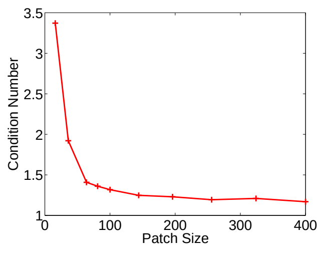
The behavior of the algorithm for the 'unstructured' Problem (P3.1) was studied as a function of the parameter λ (with $\lambda = \mu$) in our previous work [9]. It was shown that the normalized sparsification error increases with λ, while the condition number decreases. The doubly sparse transform learning algorithm for (P3.2) too has a similar behavior with respect to λ. Hence, a separate study of this parameter is omitted in this work. For natural images, the recovery PSNR using the learned transform is typically better at λ values corresponding to intermediate conditioning or 'well-conditioning', rather than unit conditioning, since unit conditioning is too restrictive [9].



(a)



(b)



(c)

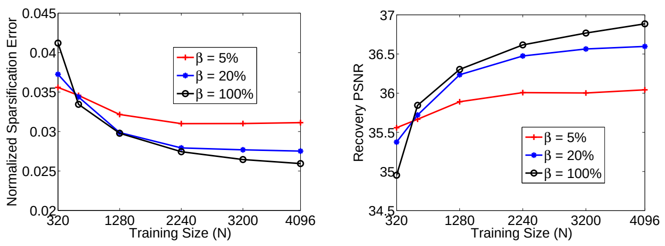
Figure 3.4: Behavior of our algorithm as function of patch size n at $\beta = 15\%$: (a) Normalized Sparsification error for the doubly sparse transform and 2D DCT, (b) Recovery PSNR for the doubly sparse transform and 2D DCT, (c) Condition number for the doubly sparse transform.

3.5.4.3 Performance as Function of Patch Size

Next, we test the performance of the algorithm for (P3.2), as a function of patch size n for the Barbara image. Non-overlapping patches are extracted at each patch size. We work with a transform sparsity level of $r = 0.15 \times n^2$, and data sparsity level of $s = 0.17 \times n$ (values rounded to nearest integers). All other parameters are fixed as in the experiment of Figure 3.2.

Figure 3.4 plots the performance metrics for both the learned doubly sparse transform and the 2D DCT, as a function of patch size. Both normalized sparsification error (Figure 3.4(a)) and recovery PSNR (Figure 3.4(b)) improve much more rapidly with increasing patch size for the learned doubly sparse transform than for the 2D DCT. In fact, the normalized sparsification error and recovery PSNR saturate for the DCT at larger patch sizes. These results indicate that doubly sparse image-specific transforms provide even better representations at larger patch sizes [8]. Furthermore, the condition

[8]We believe this effect is due to the fact that at larger patch sizes, the adapted transforms can capture high-level (or more noticeable) features of the data. On the other hand, at small patch sizes, the learning can only extract 'coarse' or low-level features. The be-



(a) (b)

Figure 3.5: Behavior of our algorithm as function of training size N at $s = 15$: (a) Normalized Sparsification error for β of 5%, 20%, and 100%, (b) Recovery PSNR for β of 5%, 20%, and 100%.

number of the learned transforms (Figure 3.4(c)) is close to 1 at all patch sizes (although the conditioning is slightly worse at low patch sizes such as $n = 16$ for the chosen parameter values). The sparsification and recovery quality improvements of the learned transform over the 2D DCT can be further increased at each patch size with optimal choice of parameters (e.g., λ).

3.5.4.4 Performance as Function of training size N

We now investigate the behavior of the algorithm for (P3.2), as a function of the size of the training set (N), at various transform sparsity fractions β. The data sparsity level is set as $s = 15$, and all other algorithm parameters except λ are fixed as in the experiment of Figure 3.3. The parameter λ is tuned at each N to enable condition numbers (for the learned transforms) similar to the experiment of Figure 3.3 (or, in other words, the $N = 4096$ case), at $s = 15$. We also select a random subset of the non-overlapping patches of the Barbara image at each N and β, for training.

Figure 3.5 plots the normalized sparsification error and recovery PSNR metrics (computed over *all* non-overlapping patches) for the learned transform as a function of training size N at various values of β. At smaller values of β, much smaller training sizes suffice to provide performance metrics similar to the $N = 4096$ case (i.e., full set of training patches). We conjecture that this is because doubly sparse transforms have far fewer free parameters, and

havior with patch size is also partly owing to the reduction in the total number of training patches with increasing n, which leads to greater adaptivity to smaller training sets.

hence, the learning requires less training signals (than unstructured transforms) to learn the optimal transform for particular data. Thus, doubly sparse transform learning can provide additional speedups in applications by utilizing smaller training sizes compared to unstructured transforms. Moreover, when N is sufficiently small, the plots of Figure 3.5 indicate that the sparsification and recovery performance of unstructured transform learning degrade much quicker than that of doubly sparse transform learning. Therefore, we conclude that the proposed doubly sparse transform learning via (P3.2) is robust to the size of the training set.

3.5.5 Global Versus Image-Specific Transforms

While we have considered adapting a transform to a specific image in the preceding experiments, here, we investigate the generalizing ability of learned transforms (which may indicate their potential for compression). In the following experiments, designed to study the generalizing power of learned transforms, we consider a magnetic resonance image dataset [91]. Of the 16 images in the dataset, 6 are used for training.

We define a 'global' transform to be one that is learned over a representative variety of training images, and tested on others. We expect such transforms to learn the characteristics common to the training set, and thus, perform well on images that share similar characteristics. We refer to this property as the generalizability of transforms. At the same time, we expect global transforms to be less adaptive to specific images than image-specific transforms. However, the question is which of the two kinds of global transforms – doubly sparse or unstructured – generalizes better?

For learning global transforms, a total of 41,334 overlapping patches of size 12×12 ($n = 144$) are extracted from the training images using a patch overlap stride of $d = 3$ [9]. The formulation of Problem (P3.2) is used to learn a doubly sparse global transform adapted to these 41,334 patches, after removing their means. The algorithm parameters are set as $\lambda = 10^7$, $s = 0.17 \times n$, and $r = 0.15 \times n^2$ (rounded to nearest integers). We also learned an unstructured global transform at r of 100%.

[9]The *overlap stride* d is defined to be the distance in pixels between corresponding pixel locations in adjacent image patches [5].

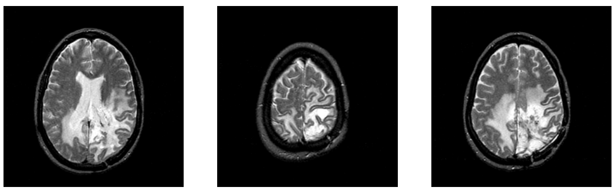


Figure 3.6: Left to Right: The three test images - R1, R2, and R3.

	CN-I	NSE-I	rPSNR-I	NSE-D	rPSNR-D	NSE-G	rPSNR-G
R1	1.15	0.0207	36.84	0.0403	33.98	0.0308	34.86
R2	1.11	0.0143	39.97	0.0347	36.11	0.0267	36.93
R3	1.14	0.0184	37.15	0.0378	34.04	0.0300	34.77

Table 3.3: Normalized sparsification errors (NSE-G) and recovery PSNRs (rPSNR-G) for the globally adapted transform at r of 15%, along with the corresponding quantities for 2D DCT (NSE-D/rPSNR-D), and for image-specific transforms (NSE-I/rPSNR-I) at r of 15%, and the condition numbers (CN-I) of the image-specific transforms.

3.5.5.1 Performance on Test Data - Doubly Sparse Versus DCT

In this experiment, we study the performance of the doubly sparse global transform on images outside the training set. We compare the performance vis-a-vis image-specific doubly sparse transforms, and the DCT.

We consider three test images (not used in training the global transform) from the MRI dataset [91], shown in Figure 3.6, and labeled as R1, R2, and R3, respectively. Table 3.3 lists the performance metrics of doubly sparse image-specific transforms of r of 15% (of n^2) and s of 17% (of n), learned on 12×12 non-overlapping patches for each of the test images. The normalized sparsification errors and recovery PSNRs obtained by applying the doubly sparse global transform on each of the test images are also listed, along with the corresponding values for the 144×144 2D DCT.

Both global and image-specific transforms (which are also well-conditioned) perform significantly better than the 2D DCT on all the test images. The doubly sparse image-specific transform provides up to 3.85 dB better normalized sparsification error and 3.86 dB better recovery PSNR than the 2D DCT. The doubly sparse global transform, although adapted to a particular training set, provides promising improvements of up to 0.88 dB in recovery PSNR and 1.17 dB in normalized sparsification error over the

	CN-I	NSE-I	rPSNR-I	NSE-G	rPSNR-G
R1	1.07	0.0127	38.94	0.0303	34.95
R2	1.04	0.0066	43.32	0.0260	37.06
R3	1.08	0.0118	39.05	0.0294	34.85

Table 3.4: Normalized sparsification errors (NSE-G) and recovery PSNRs (rPSNR-G) for the globally adapted transform at r of 100%, along with the corresponding quantities for image-specific transforms (NSE-I/rPSNR-I) at r of 100%, and the condition numbers (CN-I) of the image-specific transforms.

2D DCT on the test images. This indicates that the doubly sparse global transform is able to learn the common properties of the magnetic resonance images, and is hence able to represent the test images more effectively than analytical transforms. The results point to the promise of both the global and image-specific doubly sparse transforms for image compression. Note though that doubly sparse image-specific transforms have the additional overhead of learning and encoding/storing the learned sparse transform for each image.

3.5.5.2 Performance on Test Data - Doubly Sparse Versus Unstructured

In this experiment, we study the performance of the unstructured global transform on the test images. We compare the performance vis-a-vis the doubly sparse global transform, and image-specific unstructured transforms.

Image-specific unstructured transforms (i.e., r of 100 %) were learned for each test image. Table 3.4 lists the performance metrics of the image-specific unstructured transforms, along with the corresponding values for the unstructured global transform when applied to each of the test images.

There are several conclusions that can be drawn from the results of Table 3.4. First, the image-specific unstructured transforms perform better than the corresponding doubly sparse image-specific transforms of Table 3.3, while the latter are much cheaper to learn [10]. Second, in the case of the global transform applied to test images, the doubly sparse transform of Table 3.3 performs almost as well as an unconstrained one. Third, the results

[10]It should be noted that the performance gap between the doubly sparse and unstructured image-specific transforms tends to increase with patch size. Thus, for image-specific transforms at larger patch sizes, the choice of the analytical transform Φ may become more important for leveraging the sparsity of B.

	INSE-1	IrPSNR-1	INSE-2	IrPSNR-2	INSE-3	IrPSNR-3
T1	0.88	0.62	1.16	0.89	1.19	0.93
T2	0.87	0.56	1.13	0.82	1.47	1.16
T3	0.81	0.52	1.00	0.73	1.15	0.88

Table 3.5: The improvement in normalized sparsification error (INSE-1) and recovery PSNR (IrPSNR-1) measured in decibels for the doubly sparse global transform (at r of 15%) over 2D DCT at a patch size of $n = 100$, along with the corresponding quantities for patch sizes of $n = 144$ (INSE-2/IrPSNR-2) and $n = 196$ (INSE-3/IrPSNR-3) respectively.

also indicate a larger performance gap between the image-specific and global transforms at r of 100% than the corresponding gap at r of 15%. Apparently, reducing the number of degrees of freedom in the global transform by the sparsity constraint prevents over-fitting to a particular training image, and enables good generalization.

Computational Considerations. When a doubly sparse global transform is used in applications such as compression, the computational cost of applying the transform $B\Phi$ on a test vector is $Cr + T_\Phi$, where C is a constant, and T_Φ is the cost of applying the transform Φ (for example, $T_\Phi = O(n \log n)$ for the Hadamard, DCT, and DFT - Discrete Fourier Transform, etc.). For small r, the cost of $Cr + T_\Phi$ is much lower than the cost of applying an unstructured transform W, which scales as n^2. Moreover, the cost for the doubly sparse case involves only a small overhead above the fast analytical transform Φ.

Furthermore, the computational cost of recovering a signal/patch y from its sparse code x is also lower for the doubly sparse global transform as compared to an unstructured global transform: for sparse B, $\Phi^{-1}B^{-1}x$ can be computed efficiently, since $B^{-1}x$ can be obtained cheaply by solving a sparse system of linear equations [92, 93], and for fast transforms such as the DCT and DFT, Φ^{-1} can be applied efficiently at a cost of $O(n \log n)$.

Thus, for the global transform, the doubly sparse version is the clear winner: it provides essentially the same performance as the unstructured version, at much lower cost.

Figure 3.7: The test images used for denoising.

3.5.5.3 Performance of Global Transforms as Function of Patch Size.

Just like their image-specific counterparts, the doubly sparse global transforms perform better with increasing patch size. To illustrate this, we learn global transforms from the overlapping patches (overlap stride of $d = 3$) of the training images, at three different patch sizes of $n = 100, 144, 196$. All parameters in learning the global transforms (one at each patch size) are the same as for the global transform case of Table 3.3.

The learned global transforms at r of 15% are tested on the non-overlapping patches of the three test images of Figure 3.6, at the different patch sizes. Table 3.5 shows the improvements (measured in decibels) in normalized sparsification error and recovery PSNR for the learned transforms over the 2D DCT. The improvements over DCT increase with patch size. At $n = 196$, these improvements in recovery PSNR and normalized sparsification error are up to 1.16 dB and 1.47 dB, respectively.

3.5.6 Preliminary Results for Image Denoising

We present preliminary results for our image denoising framework (Problem (P3.4)). We consider 6 different test images (Figure 3.7) and simulate i.i.d. Gaussian noise at 5 different noise levels ($\sigma = 5, 10, 15, 20, 100$) for each of the images. All images except Cameraman (256×256) and Man (768×768) have 512×512 pixels. We use the proposed adaptive sparsifying-transform-based denoising algorithm (Section 3.4) to obtain denoised versions of the noisy images. We compare results and run times obtained by our adaptive transform-based algorithm with those obtained by adaptive overcomplete K-SVD denoising [1]. The goal is to study the potential of the proposed transform-based models (both unstructured and doubly sparse) in comparison to the dictionary-based models. The Matlab implementation of K-SVD denoising [1] available from Michael Elad's website [86] was used in our comparisons, and we used the built-in parameter settings of that implementation. Computations were performed with an Intel Core i5 CPU at 1.7GHz and 6GB memory, employing a 64-bit Windows 8 operating system.

We consider adaptive transform denoising with three different doubly sparse transforms of size 64×64, 81×81, and 121×121, each with a transform sparsity fraction $\beta = 20\%$. The corresponding patch sizes are 8×8, 9×9, and 11×11, respectively. The higher patch sizes will be used to demonstrate the scalability of doubly sparse denoising. We also learn a 64×64 unstructured transform for denoising. Since doubly sparse transforms are highly efficient, we will compare the larger doubly sparse transforms (e.g., 121×121) with the 64×64 unstructured transform, and study the trade-off between denoising and run time.

We choose Φ to be the 2D DCT in our experiments. We set $\lambda = 32 \times 10^6$ for doubly sparse transform learning (via (P3.2)), and the initial (during the first iteration of the denoising algorithm) fixed data sparsity level is $s = 0.15 \times n$ (rounded to nearest integer). In the transform update step within learning, the conjugate gradient method is run for 30 iterations. We also set a maximum iteration count in transform learning, which works together with the previously discussed stopping condition in Section 3.5.2. The maximum number of allowed iterations is lower at high noise levels, to prevent overfitting to noise. The number of initial learning iterations with no post-thresholding of the transform in the transform update step, is also set appropriately. The size of the training set used for doubly sparse learning is 22400. The training patches are chosen uniformly at random from all overlapping patches in each denoising iteration. As the patch size n increases, the fraction of all patches used in training increases. We observed that different random choices of training subsets provide denoising PSNRs that typically differ only by few hundredths of a dB.

For the unstructured transform learning case, the training set size is increased 1.43 fold compared to the doubly sparse cases[11]. Likewise, the maximum iteration count in learning is increased 2.5 fold for the unstructured case, since convergence of unstructured transform learning is slower. The parameter $\lambda = 45 \times 10^6$ ($\mu = \lambda$) for unstructured learning.

The denoising algorithm parameter $\tau = 0.01/\sigma$. The parameter $C = 1.08$ at $n = 64$, whereas it is set as 1.07 and 1.04 at $n = 81$ and $n = 121$, respectively. The denoising algorithm is executed for 4 iterations at lower noise levels ($\sigma = 5, 10$), 8 iterations at intermediate noise levels ($\sigma = 15, 20$). At very high noise ($\sigma = 100$), we use only 1 iteration, which saves run time without degrading denoising performance.

Table 3.6 lists the denoising PSNRs obtained by the doubly sparse and unstructured schemes, together with the PSNRs obtained by K-SVD. For all images and noise levels considered, the best PSNRs are provided by our transform-learning-based (either doubly sparse or unstructured) schemes. The best improvement over K-SVD averaged over all rows of Table 3.6 is 0.134 dB. Moreover, among the transform-based schemes, the best PSNRs are provided in most cases by the 121×121 doubly sparse transform [12]. The 64×64 doubly sparse transform also provides better PSNRs than K-SVD for Cameraman, Couple, and Man. For Barbara, Hill, and Lena, it provides PSNRs comparable to K-SVD in many cases. All the transforms considered have far fewer actual free parameters than the K-SVD dictionary. The 64×64 and 121×121 doubly sparse transforms have 20 fold and 5.6 fold fewer free parameters than the 64×256 K-SVD dictionary, respectively. Thus, the transform-based scheme (P3.4) provides comparable, or better PSNRs than K-SVD despite the highly constrained transforms.

Next, we compare the denoising performance of the doubly sparse transforms to the unstructured transform. The 64×64 doubly sparse transform denoises almost identically to the 64×64 unstructured transform for most images (barring Cameraman). In fact at $\sigma = 100$, it denoises slightly better than the unstructured version (on an average). Thus, the lower parametrization of the doubly sparse transforms prevents over-fitting to noise. Furthermore, on comparing the best denoising PSNRs provided by doubly sparse

[11]This training set size is still smaller than that used by K-SVD [1, 86], since the overcomplete K-SVD has more unknown parameters.

[12]We believe the larger transform denoises better, since it captures larger scale image features.

transforms of all sizes with those of the 64×64 unstructured transform (for each image and noise level), we obtain an average PSNR improvement of 0.0913 dB for the doubly sparse transforms. The 121×121 doubly sparse transform, although larger than the 64×64 unstructured transform still has fewer free parameters than the latter. Thus, doubly sparse transforms with fewer parameters provide better denoising than unstructured transforms.

Turning to Table 3.7, we can also ascertain the computational advantages of doubly sparse denoising. Table 3.7 lists the average speedups over K-SVD for the various transform-based denoising schemes. For each image and noise level, the ratio of the run times of K-SVD denoising and transform-based denoising is computed, and the speedups are averaged over the images at each noise level.

It is clear that both unstructured and doubly sparse transforms provide speedups over K-SVD. While the 64×64 unstructured transform provides a speedup of up to 4.8x over K-SVD, the speedups for the doubly sparse cases range up to 18x in Table 3.7. The speedups over K-SVD for specific images can be much higher. For example, for the Cameraman image at $\sigma = 5$, the run times for denoising with K-SVD, the 64×64 unstructured transform, 64×64 doubly sparse transform, 81×81 doubly sparse transform, and 121×121 doubly sparse transform are 15 minutes, 2.5 minutes, 35 seconds, 55 seconds, and 1.9 minutes, respectively, indicating a speedup of 25.7x for the 64×64 doubly sparse transform over K-SVD[13]. The results of Table 3.7 also indicate that the doubly sparse transform model scales well with patch size.

Now, combining the conclusions from Tables 3.6 and 3.7, we see that the 64×64 doubly sparse transform while denoising comparably to the unstructured one, is much faster (3.83x faster on average over all noise levels) than the latter. The 121×121 doubly sparse transform which denoises better than the 64×64 unstructured one, also does so more efficiently.

We show some doubly sparse denoising results in Figure 3.8, that illustrate the effectiveness of the proposed algorithm to denoise images.

All of our learned transforms are well-conditioned (condition numbers 1-3). Poorly conditioned transforms were empirically observed to degrade denoising performance. Although we presented denoising results at 20% transform sparsity, the denoising performances are reasonable even at lower transform

[13]We anticipate greater speedups with optimized implementation of sparse matrix operations.

sparsities such as 10%, where the speedups over previous methods is even greater.

Comparison to Other Methods.

The recent overcomplete analysis-dictionary-based image denoising method of Yaghoobi et al. [70] was shown by the authors to denoise worse than K-SVD. Hence, we have only included the comparison to K-SVD here.

A method closely related to our proposed approach is the synthesis double sparsity method of Rubinstein et al. [60]. However, for 2D image denoising, it was shown by the authors in [60] (also confirmed in our own experiments with the author's code [94]) that the synthesis double sparsity method, although faster than standard K-SVD [1], typically falls behind in denoising performance. Hence, we only compare to the (better) unconstrained K-SVD [1] in this work. Synthesis double sparsity has been shown to be more beneficial for 3D applications such as 3D denoising [60]. We plan to study such applications in the near future. Note that the speedups over K-SVD shown in Table 3.7 for the 2D image cases in this work, can only increase, upon switching to 3D data. This is because the learning via K-SVD is significantly slower than transform learning (Section 3.3.3) at larger problem sizes [9].

The denoising PSNRs presented for our algorithm can be further improved by optimal choice of parameters. Our experiments thus demonstrate the promise of learned doubly sparse transforms in image denoising. The results with just the adaptive square transforms are comparable to, or better than with overcomplete dictionaries. Doubly sparse transforms denoise as well or better than unstructured transforms, but do so much faster.

We emphasize that the transforms considered in this work are only square transforms. We conjecture that the denoising performance of our algorithms would improve further and become comparable to the state of the art (for example [7]) with overcomplete and multiscale extensions of transform learning (similarly to the synthesis case [59]).

3.6 Conclusions

In this chapter, we proposed novel problem formulations for learning square doubly sparse transforms. Our formulations give rise to significantly sparse

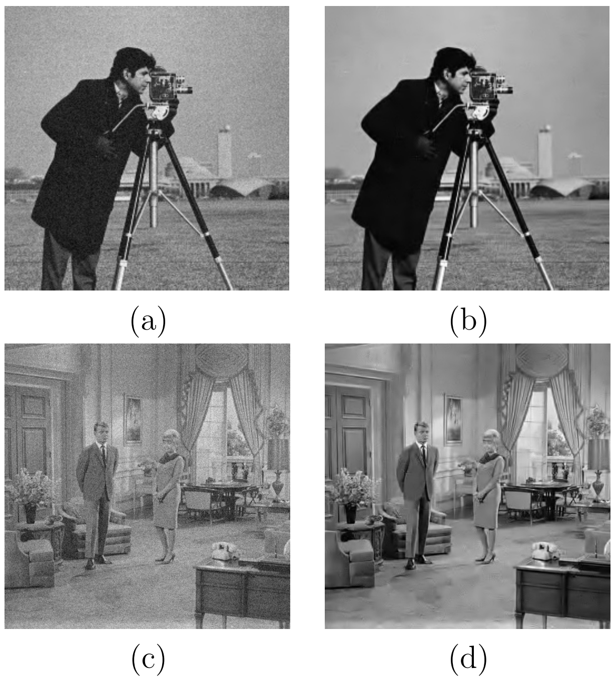


Figure 3.8: Denoising: (a) Noisy ($\sigma = 10$) cameraman image (PSNR = 28.14 dB), (b) Denoised cameraman image (PSNR = 33.81 dB) using 64×64 transform at $\beta = 20\%$, (c) Noisy ($\sigma = 20$) couple image (PSNR = 22.11 dB), (d) Denoised couple image (PSNR = 30.14 dB) using 121×121 transform at $\beta = 20\%$.

and well-conditioned transforms with better sparsification errors and recovery PSNRs than analytical transforms. Moreover, imposing the doubly sparse property leads to faster learning and faster computations with the sparse transform. The adapted doubly sparse transform has reduced storage requirements and generalizes better than the non-sparse transform. We also discussed a novel problem formulation for image denoising in this chapter, and demonstrated the promise of adaptive sparsifying transforms in this application, with results competitive with overcomplete (synthesis) K-SVD. Importantly, denoising with doubly sparse transforms is computationally very cheap.

Image	σ	K-SVD	Unstr-uctured	Doubly Sparse		
				$n = 64$	$n = 81$	$n = 121$
1	5	38.08	38.16	38.14	38.20	**38.25**
	10	34.41	34.36	34.37	34.43	**34.53**
	15	32.33	32.09	32.06	32.20	**32.37**
	20	30.83	30.53	30.53	30.71	**30.90**
	100	21.87	21.91	21.91	22.06	**22.26**
2	5	37.81	**38.04**	37.91	37.86	37.82
	10	33.72	**33.91**	33.81	33.81	33.80
	15	31.50	**31.68**	31.58	31.61	31.61
	20	29.82	29.90	29.83	29.86	**29.91**
	100	21.76	21.79	21.82	21.93	**22.06**
3	5	37.28	**37.33**	37.30	37.30	37.30
	10	33.51	33.62	33.59	33.61	**33.64**
	15	31.46	31.51	31.47	31.50	**31.59**
	20	30.02	30.03	30.02	30.04	**30.14**
	100	22.57	22.58	22.59	22.69	**22.78**
4	5	36.47	**36.68**	36.64	36.64	36.64
	10	32.71	**32.97**	32.92	32.92	32.93
	15	30.78	30.98	30.96	30.96	**30.99**
	20	29.40	29.57	29.54	29.55	**29.60**
	100	22.76	22.89	22.90	22.99	**23.01**
5	5	37.08	**37.10**	37.04	37.04	37.08
	10	33.45	33.43	33.40	33.44	**33.49**
	15	31.52	31.49	31.47	31.50	**31.57**
	20	30.17	30.10	30.09	30.13	**30.23**
	100	23.98	23.81	23.79	23.94	**24.00**
6	5	38.61	**38.64**	38.57	38.60	38.63
	10	35.49	35.52	35.47	35.52	**35.56**
	15	33.74	33.69	33.67	33.75	**33.80**
	20	32.41	32.27	32.27	32.35	**32.44**
	100	24.51	24.31	24.29	24.49	**24.60**

Table 3.6: PSNR values for denoising for our algorithm using 64×64, 81×81, and 121×121 doubly sparse transforms at $\beta = 20\%$. Our results are compared with those obtained using a 64×64 unstructured transform, and a 64×256 overcomplete K-SVD dictionary [1]. The best PSNRs are marked in bold. The images numbered 1 through 6 correspond to Barbara, Cameraman, Couple, Man, Hill, and Lena, respectively.

σ	Unstructured $(n = 64)$	Doubly Sparse $(n = 64)$	Doubly Sparse $(n = 81)$	Doubly Sparse $(n = 121)$
5	4.8	18.0	11.6	5.6
10	3.3	13.8	9.1	4.3
15	1.7	7.5	5.0	2.3
20	1.2	5.6	3.7	1.7
100	3.0	6.4	4.9	3.0

Table 3.7: The denoising speedups over K-SVD [1] for the various transform learning schemes. The speedups are averaged over the six images at each noise level.

CHAPTER 4

SQUARE TRANSFORM LEARNING WITH OPTIMAL UPDATES AND CONVERGENCE GUARANTEES

4.1 Introduction

Focusing on the transform model, we developed formulations and algorithms for the learning of well-conditioned sparsifying transforms in Chapter 2. In that chapter, as well as here, we restrict ourselves to square transform matrices. In this follow-on work, we derive highly *efficient closed-form solutions* for the update steps in transform learning, that further enhance the convergence and computational properties of our alternating learning algorithms [1]. Importantly, we establish that our iterative algorithms are globally convergent to the set of local minimizers of the non-convex transform learning problems.

We organize the rest of this chapter as follows. Section 4.2 briefly recalls the transform learning formulation that is analyzed in this chapter. In Section 4.3, we derive efficient algorithms for transform learning, and discuss the algorithms' computational cost. In Section 4.4, we present convergence guarantees for our algorithms. The proof of convergence is provided in Appendix B. Section 4.5 presents experimental results demonstrating the convergence behavior, and the computational efficiency of the proposed scheme. We also show brief results for the image denoising application. In Section 4.6, we conclude.

4.2 Problem Formulation

The formulation (P2.3) for unstructured square transform learning was presented in Chapter 2. Here, we only briefly recall the formulation and state

[1]The material of this chapter has been recently presented in [6], [95], and [96].

the key properties relevant to this chapter.

Given a matrix $Y \in \mathbb{R}^{n \times N}$, whose columns represent training signals, the transform learning formulation is

$$\text{(P4.0)} \quad \min_{W,X} \|WY - X\|_F^2 + \lambda \left(\xi \|W\|_F^2 - \log |\det W| \right) \\ s.t. \;\; \|X_i\|_0 \leq s \;\; \forall \; i$$

where $\lambda > 0$, $\xi > 0$ are parameters, $W \in \mathbb{R}^{n \times n}$ is the square transform, and $X \in \mathbb{R}^{n \times N}$ is the matrix, whose columns X_i are the sparse codes of the corresponding training signals Y_i. Similarly as in Chapter 3, we work with the absolute value of the determinant in the formulation above.

In this chapter, we let $v(W) \triangleq -\log |\det W| + \xi \|W\|_F^2$ denote the regularizer in Problem (P4.0). The various properties of formulation (P2.3) mentioned in Chapter 2 trivially hold/extend with the introduction of the absolute value for the determinant in (P4.0). Recall that we can always switch from a W with $\det W < 0$ to one with $\det W > 0$ trivially by swapping two rows of W. The regularizer $v(W)$ prevents trivial solutions in (P4.0), and controls the condition number $\kappa(W)$ and scaling of the learned transform. Badly conditioned transforms typically convey little information and may degrade performance in applications such as signal/image representation, and denoising. The condition number $\kappa(W)$ is upper bounded by a monotonically increasing function of $v(W)$. Hence, minimizing $v(W)$ encourages reduction of the condition number. For a fixed ξ, as $\lambda \to \infty$ in (P4.0), the condition number of the optimal transform(s) tends to 1, and their spectral norm (or, scaling) tends to $1/\sqrt{2\xi}$. Specifically, for $\xi = 0.5$, as $\lambda \to \infty$, the optimal transform tends to an *orthonormal transform*.

In this work, to achieve invariance of the learned transform to trivial scaling of the training data Y, we set $\lambda = \lambda_0 \|Y\|_F^2$ in (P4.0), where $\lambda_0 > 0$ is a constant. Indeed, when the data Y are replaced with αY ($\alpha \in \mathbb{R}$, $\alpha \neq 0$) in (P4.0), we can set $X = \alpha X'$. Then, the objective function becomes $\alpha^2 \left(\|WY - X'\|_F^2 + \lambda_0 \|Y\|_F^2 \, v(W) \right)$, which is just a scaled version of the objective in (P4.0) (for un-scaled Y). Hence, its minimization over (W, X') (with X' constrained to have columns of sparsity $\leq s$) yields the same solution(s) as (P4.0). Thus, the learned transform for data αY is the same as for Y, while the learned sparse code for αY is α times that for Y.

The cost function in (P4.0) is lower bounded by λv_0, where $v_0 = \frac{n}{2} + \frac{n}{2}\log(2\xi)$ (cf. Chapter 2). The minimum objective value in Problem (P4.0) equals the bound λv_0 if and only if there exists a pair $(\hat{W}, \hat{X})$ such that $\hat{W}Y = \hat{X}$, with $\hat{X} \in \mathbb{R}^{n \times N}$ whose columns have sparsity $\leq s$, and $\hat{W} \in \mathbb{R}^{n \times n}$ with $\kappa(\hat{W}) = 1$. Thus, when an "error-free" transform model exists for the data, and the underlying transform is unit conditioned, such a transform model (up to scaling) is guaranteed to be a global minimizer of Problem (P4.0) (i.e., such a model is *identifiable* by solving (P4.0)).

Finally, because the objective in (P4.0) is unitarily invariant, then given a minimizer $(\tilde{W}, \tilde{X})$, the pair $(\Theta\tilde{W}, \Theta\tilde{X})$ is another equivalent minimizer for all *sparsity-preserving orthonormal matrices* Θ, i.e., Θ such that $\|\Theta\tilde{X}_i\|_0 \leq s$ $\forall\, i$. For example, Θ can be a row permutation matrix, or a diagonal ± 1 sign matrix.

4.3 Transform Learning Algorithm

4.3.1 Algorithm

We have previously proposed an alternating algorithm for solving (P4.0) in Chapter 2, that alternates between solving for X (*sparse coding step*) and W (*transform update step*), with the other variable kept fixed. While the sparse coding step has an exact solution, the transform update step was performed using iterative nonlinear conjugate gradients (NLCG). This alternating algorithm for transform learning has a low computational cost compared to synthesis/analysis dictionary learning. In the following, we provide a further improvement: we show that instead, both steps of transform learning can in fact, be performed exactly and cheaply.

4.3.1.1 Sparse Coding Step

The sparse coding step in the alternating algorithm for (P4.0) is as follows [9]:

$$\min_{X} \|WY - X\|_F^2 \quad s.t. \quad \|X_i\|_0 \leq s \;\; \forall\, i \tag{4.1}$$

The above problem is to project WY onto the (non-convex) set of matrices whose columns have sparsity $\leq s$. Due to the additivity of the objective, this corresponds to projecting each column of WY onto the set of sparse vectors $\{x \in \mathbb{R}^n : \|x\|_0 \leq s\}$, which we call the s-ℓ_0 ball. Now, for a vector $z \in \mathbb{R}^n$, the optimal projection $\hat{z}$ onto the s-ℓ_0 ball is computed by zeroing out all but the s coefficients of largest magnitude in z. If there is more than one choice for the s coefficients of largest magnitude in z (can occur when multiple entries in z have identical magnitude), then the optimal $\hat{z}$ is not unique. We then choose $\hat{z} = H_s(z)$, where $H_s(z)$ is the projection, for which the indices of the s largest magnitude elements (in z) are the lowest possible. Hence, an optimal sparse code in (4.1) is computed as $\hat{X}_i = H_s(WY_i)\ \forall\, i$.

Alternatively, if the ℓ_0 sparsity constraints in (P4.0) are replaced with ℓ_0 penalties in the objective (this version of the transform learning problem has been used for example in adaptive tomographic reconstruction [97, 98]), we solve the following sparse coding problem:

$$\min_{X} \|WY - X\|_F^2 + \sum_{i=1}^{N} \eta_i^2 \|X_i\|_0 \tag{4.2}$$

where η_i^2 (with $\eta_i > 0\ \forall\, i$) denote the weights (e.g., $\eta_i = \eta\ \forall\, i$ for some η) for the sparsity penalties. A solution $\hat{X}$ of (4.2) in this case is obtained as $\hat{X}_i = \hat{H}^1_{\eta_i}(WY_i)\ \forall\, i$, where the (hard-thresholding) operator $\hat{H}^1_\eta(\cdot)$ is defined as

$$\left(\hat{H}^1_\eta(b)\right)_j = \begin{cases} 0 & , \ |b_j| < \eta \\ b_j & , \ |b_j| \geq \eta \end{cases} \tag{4.3}$$

Here, $b \in \mathbb{R}^n$, and the subscript j indexes vector entries. When the condition $|(WY)_{ji}| = \eta_i$ occurs for some i and j (where $(WY)_{ji}$ is the element of WY on the j^{th} row and i^{th} column), the corresponding optimal $\hat{X}_{ji}$ can be either $(WY)_{ji}$ or 0 (both of which correspond to the minimum value of the cost in (4.2)). The definition in (4.3) breaks the tie between these equally valid solutions by selecting the first. Thus, similar to Problem (4.1), the solution to (4.2) can be computed exactly.

4.3.1.2 Transform Update Step

The transform update step of (P4.0) involves the following unconstrained non-convex [9] minimization.

$$\min_{W} \; \|WY - X\|_F^2 + \lambda\xi \, \|W\|_F^2 - \lambda \log \, |\det W| \tag{4.4}$$

Note that although NLCG works well for the transform update step [9], convergence to the global minimum of the non-convex transform update step has not been proved with NLCG. Instead, replacing NLCG, the following proposition provides the closed-form solution for Problem (4.4). The solution is written in terms of an appropriate singular value decomposition (SVD). We use $(\cdot)^T$ to denote the matrix transpose operation, and $M^{\frac{1}{2}}$ to denote the positive definite square root of a positive definite matrix M. We let I denote the $n \times n$ identity matrix.

Proposition 2. *Given the training data $Y \in \mathbb{R}^{n\times N}$, sparse code matrix $X \in \mathbb{R}^{n\times N}$, and $\lambda > 0$, $\xi > 0$, factorize $YY^T + \lambda\xi I$ as LL^T, with $L \in \mathbb{R}^{n\times n}$. Further, let $L^{-1}YX^T$ have a full SVD of $Q\Sigma R^T$. Then, a global minimizer for the transform update step* (4.4) *can be written as*

$$\hat{W} = 0.5R\left(\Sigma + \left(\Sigma^2 + 2\lambda I\right)^{\frac{1}{2}}\right) Q^T L^{-1} \tag{4.5}$$

The solution is unique if and only if $L^{-1}YX^T$ is non-singular. Furthermore, the solution is invariant to the choice of factor L.

Proof: The objective function in (4.4) can be re-written as $tr\left\{W\left(YY^T + \lambda\xi I\right)W^T\right\} - 2\,tr(WYX^T) + tr(XX^T) - \lambda \log\,|\det W|$. We then decompose the positive-definite matrix $YY^T + \lambda\xi I$ as LL^T (e.g., L can be the positive-definite square root, or the Cholesky factor of $YY^T + \lambda\xi I$). The objective function then simplifies as follows

$$tr\left(WLL^TW^T - 2WYX^T + XX^T\right) - \lambda \log\,|\det W|$$

Using a change of variables $B = WL$, the multiplicativity of the determinant implies $\log\,|\det B| = \log\,|\det W| + \log\,|\det L|$. Problem (4.4) is then equivalent to

$$\min_{B} \; tr\left(BB^T\right) - 2tr\left(BL^{-1}YX^T\right) - \lambda \log\,|\det B| \tag{4.6}$$

Next, let $B = U\Gamma V^T$, and $L^{-1}YX^T = Q\Sigma R^T$ be full SVDs ($U, \Gamma, V, Q, \Sigma, R$ are all $n \times n$ matrices), with γ_i and σ_i denoting the diagonal entries of Γ and Σ, respectively. The unconstrained minimization (4.6) then becomes

$$\min_{\Gamma} \left[tr\left(\Gamma^2\right) - 2 \max_{U,V} \left\{ tr\left(U\Gamma V^T Q\Sigma R^T\right) \right\} - \lambda \sum_{i=1}^{n} \log\, \gamma_i \right]$$

For the inner maximization, we use the inequality $tr\left(U\Gamma V^T Q\Sigma R^T\right) \leq tr\left(\Gamma\Sigma\right)$ [99], with the upper bound being attained by setting $U = R$ and $V = Q$. The remaining minimization with respect to Γ is then

$$\min_{\{\gamma_i\}} \sum_{i=1}^{n} \gamma_i^2 - 2\sum_{i=1}^{n} \gamma_i\sigma_i - \lambda \sum_{i=1}^{n} \log\, \gamma_i \tag{4.7}$$

This problem is convex in the non-negative singular values γ_i, and the solution is obtained by differentiating the cost in (4.7) with respect to the γ_i's and setting the derivative to 0. This gives $\gamma_i = 0.5\left(\sigma_i \pm \sqrt{\sigma_i^2 + 2\lambda}\right)\ \forall\, i$. Since all the $\gamma_i \geq 0$, the only feasible solution is

$$\gamma_i = \frac{\sigma_i + \sqrt{\sigma_i^2 + 2\lambda}}{2}\ \forall\, i \tag{4.8}$$

Thus, a closed-form solution or global minimizer for the transform update step (4.4) is given as in (4.5).

The solution (4.5) is invariant to the specific choice of the matrix L. To show this, we will first show that if $L_1 \in \mathbb{R}^{n\times n}$ and $L_2 \in \mathbb{R}^{n\times n}$ satisfy $YY^T + \lambda\xi I = L_1L_1^T = L_2L_2^T$, then $L_2 = L_1G$, where G is an orthonormal matrix satisfying $GG^T = I$. A brief proof of the latter result is as follows. Since L_1 and L_2 are both $n \times n$ full rank matrices (being square roots of the positive definite matrix $YY^T + \lambda\xi I$), we have $L_2 = L_1\left(L_1^{-1}L_2\right) = L_1G$, with $G \triangleq L_1^{-1}L_2$ a full rank matrix. Moreover, since $L_1L_1^T - L_2L_2^T = 0$, we have

$$L_1\left(I - GG^T\right)L_1^T = 0 \tag{4.9}$$

Because L_1 has full rank, we must therefore have that $GG^T = I$ for (4.9) to hold. Therefore, G is an orthonormal matrix satisfying $GG^T = I$.

Consider L_1 and L_2 as defined above. Now, if Q_1 is the left singular matrix corresponding to $L_1^{-1}YX^T$, then $Q_2 = G^TQ_1$ is a corresponding left singular matrix for $L_2^{-1}YX^T = G^T L_1^{-1}YX^T$. Therefore, replacing L_1 by L_2 in (4.5), making the substitutions $L_2^{-1} = G^T L_1^{-1}$, $Q_2^T = Q_1^T G$, and using the orthonormality of G, it is obvious that the closed-form solution (4.5) involving L_2 is identical to that involving L_1.

Finally, we show that the solution (4.5) is unique if and only if $L^{-1}YX^T$ is non-singular. First, the solution (4.5) can be written using the notations introduced above as $\hat{W} = \left(\sum_{i=1}^{n} \gamma_i R_i Q_i^T\right) L^{-1}$, where R_i and Q_i are the i$^{\text{th}}$ columns of R and Q, respectively. We first show that the non-singularity of $L^{-1}YX^T$ is a necessary condition for uniqueness of the solution to (4.4). Now, if $L^{-1}YX^T$ has rank $< n$, then a singular vector pair $\left(Q_k, R_k\right)$ of $L^{-1}YX^T$ corresponding to a zero singular value (say $\sigma_k = 0$) can also be modified as $\left(Q_k, -R_k\right)$ or $\left(-Q_k, R_k\right)$, yielding equally valid alternative SVDs of $L^{-1}YX^T$. However, because zero singular values in the matrix Σ are mapped to non-zero singular values in the matrix Γ (by (4.8)), we have that the following two matrices are equally valid solutions to (4.4).

$$\hat{W}^a = \left(\sum\nolimits_{i \neq k} \gamma_i R_i Q_i^T + \gamma_k R_k Q_k^T\right) L^{-1} \tag{4.10}$$

$$\hat{W}^b = \left(\sum\nolimits_{i \neq k} \gamma_i R_i Q_i^T - \gamma_k R_k Q_k^T\right) L^{-1} \tag{4.11}$$

where $\gamma_k > 0$. It is obvious that $\hat{W}^a \neq \hat{W}^b$, i.e., the optimal transform is not unique in this case. Therefore, $L^{-1}YX^T$ being non-singular is a necessary condition for uniqueness of the solution to (4.4).

Next, we show that the non-singularity of $L^{-1}YX^T$ is also a sufficient condition for the aforementioned uniqueness. First, if the singular values of $L^{-1}YX^T$ are non-degenerate (distinct and non-zero), then the SVD of $L^{-1}YX^T$ is unique up to joint scaling of any pair $\left(Q_i, R_i\right)$ by ± 1. This immediately implies that the solution $\hat{W} = \left(\sum_{i=1}^{n} \gamma_i R_i Q_i^T\right) L^{-1}$ is unique in this case. On the other hand, if $L^{-1}YX^T$ has some repeated but still non-zero singular values, then they are mapped to repeated (and non-zero) singular values in Γ by (4.8). Let us assume that Σ has only one singular value that repeats (the proof easily extends to the case of multiple repeated singular values) say r times, and that these repeated values are arranged in the bottom half of the matrix Σ (i.e., $\sigma_{n-r+1} = \sigma_{n-r+2} = ... = \sigma_n = \hat{\sigma} > 0$).

Then, we have

$$L^{-1}YX^T = \sum_{i=1}^{n-r} \sigma_i Q_i R_i^T + \hat{\sigma} \left(\sum_{i=n-r+1}^{n} Q_i R_i^T \right) \tag{4.12}$$

Because the matrix defined by the first sum on the right (in (4.12)) corresponding to distinct singular values is unique, and $\hat{\sigma} > 0$, so too is the second matrix defined by the second sum. (This is also a simple consequence of the fact that although the singular vectors associated with repeated singular values are not unique, the subspaces spanned by them are.) The transform update solution (4.5) in this case is given as

$$\hat{W} = \left\{ \sum_{i=1}^{n-r} \gamma_i R_i Q_i^T + \gamma_{n-r+1} \left(\sum_{i=n-r+1}^{n} R_i Q_i^T \right) \right\} L^{-1} \tag{4.13}$$

Based on the preceding arguments, it is clear that the right hand side of (4.13) is unique, irrespective of the particular choice of (non-unique) Q and R. Thus, the transform update solution (4.5) is unique when $L^{-1}YX^T$ has possibly repeated, but non-zero singular values. Therefore, the non-singularity of $L^{-1}YX^T$ is also a sufficient condition for the uniqueness of the solution to (4.4). $\blacksquare$

The transform update solution (4.5) is expressed in terms of the full SVD of $L^{-1}YX^T$, where L is for example, the eigenvalue decomposition (EVD) square root of $YY^T + \lambda\xi I$. Although in practice the SVD, the EVD, and even the square root of non-negative scalars, are computed using iterative methods, we will assume in the theoretical analysis in this chapter, that the solution (4.5) is computed exactly. In practice, standard numerical methods are guaranteed to quickly provide machine precision accuracy (the best practical accuracy) for the SVD (as well as the EVD, and scalar square root). Therefore, the transform update solution (4.5) is computed to within machine precision accuracy in practice.

The Algorithms A1 and A2 (corresponding to (4.1) and (4.2), respectively) for transform learning are shown in Fig. 4.1. Although we begin with the transform update step in each iteration in Fig. 4.1, one could alternatively start with the sparse coding step as well.

While Proposition 2 provides the closed-form solution to equation (4.4) for real-valued matrices, the solution can be extended to the complex-valued case (useful in applications such as magnetic resonance imaging (MRI) [5]) by

```
Transform Learning Algorithms A1 and A2
**Input :** $Y$ - training data, $s$ - sparsity, $\lambda$ - constant, $\xi$ - constant, $J_0$ - number of iterations.
**Output :** $\hat{W}$ - learned transform, $\hat{X}$ - learned sparse code matrix.
**Initial Estimates:** $(W^0, X^0)$.
**Pre-Compute:** $L^{-1} = \left(YY^T + \lambda\xi I\right)^{-1/2}$.
**For k = 1: J_0 Repeat**
    1. Compute full SVD of $L^{-1}Y(\hat{X}^{k-1})^T$ as $Q\Sigma R^T$.
    2. $\hat{W}^k = 0.5R\left(\Sigma + (\Sigma^2 + 2\lambda I)^{\frac{1}{2}}\right)Q^T L^{-1}$.
    3. $\hat{X}_i^k = H_s(W^k Y_i)\ \forall\, i$ for Algorithm A1, or $\hat{X}_i^k = \hat{H}_{\eta_i}^1(W^k Y_i)\ \forall\, i$ for Algorithm A2.
**End**
```

Figure 4.1: Algorithms A1 and A2 for learning the transform and sparse code. A superscript of k is used to denote the iterates in the algorithms. Although we begin with the transform update step in each iteration above, one could alternatively start with the sparse coding step as well.

replacing the $(\cdot)^T$ operation in Proposition 2 and its proof by $(\cdot)^H$, the Hermitian transpose operation. The same proof applies, with the trace bound for the real case replaced by $Re\left\{tr\left(U\Gamma V^H Q\Sigma R^H\right)\right\} \leq tr\left(\Gamma\Sigma\right)$ for the complex case, where $Re(A)$ denotes the real part of scalar A.

4.3.2 The Orthonormal Transform Limit

We have seen that for $\xi = 0.5$, as $\lambda \to \infty$, the W minimizing (P4.0) tends to an orthonormal matrix. Here, we study the behavior of the actual sparse coding and transform update steps of our algorithm as the parameter λ (or, equivalently λ_0, since $\lambda = \lambda_0 \|Y\|_F^2$) tends to infinity. Proposition 3 establishes that as $\lambda \to \infty$ with ξ held at 0.5, the sparse coding and transform update solutions for (P4.0) approach the corresponding solutions for an orthonormal transform learning problem.

Proposition 3. *For $\xi = 0.5$, as $\lambda \to \infty$, the sparse coding and transform update solutions in (P4.0) coincide with the corresponding solutions obtained by employing alternating minimization on the following orthonormal transform* *learning problem.*

$$\min_{W,X} \|WY - X\|_F^2 \;\; s.t. \;\; W^T W = I, \; \|X_i\|_0 \leq s \;\; \forall \; i \tag{4.14}$$

Specifically, the sparse coding step for Problem (4.14) *involves*

$$\min_{X} \|WY - X\|_F^2 \;\; s.t. \;\; \|X_i\|_0 \leq s \;\; \forall \; i$$

and the solution is $\hat{X}_i = H_s(WY_i) \; \forall \, i$. *Moreover, the transform update step for Problem* (4.14) *involves*

$$\max_{W} tr\left(WYX^T\right) \;\; s.t. \;\; W^T W = I \tag{4.15}$$

Denoting the full SVD of YX^T *by* $U\Sigma V^T$, *where* $U \in \mathbb{R}^{n\times n}$, $\Sigma \in \mathbb{R}^{n\times n}$, $V \in \mathbb{R}^{n\times n}$, *the optimal solution in Problem* (4.15) *is* $\hat{W} = VU^T$. *This solution is unique if and only if the singular values of* YX^T *are non-zero.*

For $\xi \neq 0.5$, Proposition 3 holds with the constraint $W^T W = I$ in Problem (4.14) replaced by the constraint $W^T W = (1/2\xi)I$. The transform update solution for Problem (4.14) with the modified constraint $W^T W = (1/2\xi)I$ is the same as mentioned in Proposition 3, except for an additional scaling of $1/\sqrt{2\xi}$. The proof of Proposition 3 is provided in Appendix B.1.

The orthonormal transform case is special, in that Problem (4.14) is also an orthonormal synthesis dictionary learning problem, with W^T denoting the synthesis dictionary. This follows immediately, using the identity $\|WY - X\|_F = \|Y - W^T X\|_F$, for orthonormal W. Hence, Proposition 3 provides an alternating algorithm with optimal updates not only for the orthonormal transform learning problem, but at the same time for the orthonormal dictionary learning problem.

4.3.3 Computational Cost

The proposed transform learning algorithms A1 and A2 alternate between the sparse coding and transform update steps. Each of these steps has a closed-form solution. We now discuss their computational costs. We assume that the matrices $YY^T + \lambda I$ and L^{-1} (used in (4.5)) are pre-computed (at the beginning of the algorithm) at total costs of $O(Nn^2)$ and $O(n^3)$, respectively, for the entire algorithm.

The computational cost of the sparse coding step in both Algorithms A1 and A2 is dominated by the computation of the product WY, and therefore scales as $O(Nn^2)$. In contrast, the projection onto the s-ℓ_0 ball in Algorithm A1 requires only $O(nN \log n)$ operations, when employing sorting [9], and the hard thresholding (as in equation (4.3)) in Algorithm A2 requires only $O(nN)$ comparisons.

For the transform update step, the computation of the product YX^T requires αNn^2 multiply-add operations for an X with s-sparse columns, and $s = \alpha n$. Then, the computation of $L^{-1}YX^T$, its SVD, and the closed-form transform update (4.5) require $O(n^3)$ operations. On the other hand, when NLCG is employed for transform update, the cost (excluding the YX^T pre-computation) scales as $O(Jn^3)$, where J is the number of NLCG iterations [9]. Thus, compared to NLCG, the proposed update formula (4.5) allows for both an exact and potentially cheap (depending on J) solution to the transform update step.

Under the assumption that $n \ll N$, the total cost per iteration (of sparse coding and transform update) of the proposed algorithms scales as $O(Nn^2)$. This is much lower than the per-iteration cost of learning an $n \times K$ overcomplete ($K > n$) synthesis dictionary D using K-SVD [17], which scales (assuming that the synthesis sparsity level $s \propto n$) as $O(KNn^2)$. Our transform learning schemes also hold a similar computational advantage over analysis dictionary learning schemes such as analysis K-SVD [9].

As illustrated in Section 4.5.2, our algorithms converge in few iterations in practice. Therefore, the per-iteration computational advantages (e.g., over K-SVD) also typically translate to a net computational advantage in practice (e.g., in denoising).

4.4 Main Convergence Results

4.4.1 Result for Problem (P4.0)

Problem (P4.0) has the constraint $\|X_i\|_0 \leq s \, \forall \, i$, which can instead be added as a penalty in the objective by using a barrier function $\psi(X)$ (which takes the value $+\infty$ when the constraint is violated, and is zero otherwise). In this form, Problem (P4.0) is unconstrained, and we denote its objective as $g(W, X) = \|WY - X\|_F^2 + \lambda\xi \|W\|_F^2 - \lambda \log |\det W| + \psi(X)$. The unconstrained minimization problem involving the objective $g(W, X)$ is exactly equivalent to the constrained formulation (P4.0) in the sense that the minimum objective values as well as the set of minimizers of the two formulations are identical. To see this, note that whenever the constraint $\|X_i\|_0 \leq s \, \forall \, i$ is satisfied, the two objectives coincide. Otherwise, the objective in the unconstrained formulation takes the value $+\infty$ and therefore, its minimum value is achieved where the constraint $\|X_i\|_0 \leq s \, \forall \, i$ holds. This minimum value (and the corresponding set of minimizers) is therefore the same as that for the constrained formulation (P4.0). The proposed Algorithm A1 is an exact alternating algorithm for both the constrained and unconstrained formulations above.

Problem (P4.0) is to find the best possible transform model for the given training data Y by minimizing the sparsification error, and controlling the condition number (avoiding triviality). We are interested to know whether the proposed alternating algorithm converges to a minimizer of (P4.0), or whether it could get stuck in saddle points, or some non-stationary points. Problem (P4.0) is non-convex, and therefore, well-known results on convergence of alternating minimization (e.g., [100]) do not apply here. Since the sparse coding and transform update steps in Algorithm A1 have elegant closed-form solutions, we utilize these solutions in our analysis to prove convergence. The following Theorem 1 provides the convergence of our Algorithm A1 for (P4.0). We say that a sequence $\{a^k\}$ has an accumulation point a, if there is a subsequence that converges to a. For a vector h, we let $\phi_j(h)$ denote the magnitude of the j^{th} largest element (magnitude-wise) of h. For some matrix B, $\|B\|_\infty \triangleq \max_{i,j} |B_{ij}|$ [2].

Theorem 1. *Let $\{W^k, X^k\}$ denote the iterate sequence generated by Algorithm A1 with training data Y and initial (W^0, X^0). Then, the objective sequence $\{g(W^k, X^k)\}$ is monotone decreasing, and converges to a finite value, say $g^* = g^*(W^0, X^0)$. Moreover, the iterate sequence is bounded, and any specific accumulation point (W, X) of the iterate sequence is a fixed point of*

[2]This is a non-standard definition (compared to Horn and Johnson [101]) of the infinity norm of a matrix.

the algorithm satisfying the following local optimality condition.

$$g(W + dW, X + \Delta X) \geq g(W, X) = g^* \tag{4.16}$$

The condition holds for all sufficiently small $dW \in \mathbb{R}^{n \times n}$ *satisfying* $\|dW\|_F \leq \epsilon'$ *for some* $\epsilon' > 0$ *that depends on the specific* W*, and all* $\Delta X \in \mathbb{R}^{n \times N}$ *in the union of the following regions.*

R1. The half-space tr $\left\{(WY - X)\Delta X^T\right\} \leq 0$.

R2. The local region defined by
$\|\Delta X\|_\infty < \min_i \left\{\phi_s(WY_i) : \|WY_i\|_0 > s\right\}$.

Furthermore, if we have $\|WY_i\|_0 \leq s \, \forall \, i$*, then* ΔX *can be arbitrary.*

Local region R2 above is defined in terms of the scalar $\min_i \left\{\phi_s(WY_i) : \|WY_i\|_0 > s\right\}$, which is computed by taking the columns of WY with sparsity $> s$, and finding the s-largest magnitude element in each of these columns, and choosing the smallest of those magnitudes. The intuition for this particular construction of the local region is provided in the proof of Lemma 23 in Appendix B.3.

Theorem 1 indicates local convergence of our alternating Algorithm A1. Assuming a particular initial (W^0, X^0), we have that every accumulation point (W, X) of the iterate sequence is a local optimum by equation (4.16), and satisfies $g(W, X) = g^*\left(W^0, X^0\right)$. (Note that g^* depends on only the initial (W^0, X^0).) Thus, all accumulation points of the iterates (for a particular initial (W^0, X^0)) are equivalent (in terms of their cost), or are equally good local minima. We thus have the following corollary.

Corollary 3. *For Algorithm A1, assuming a particular initial* (W^0, X^0)*, the objective converges to a local minimum, and the iterates converge to an equivalence class of local minimizers.*

The local optimality condition (4.16) holds for the algorithm irrespective of initialization. However, the local minimum $g^*\left(W^0, X^0\right)$ that the objective converges to may possibly depend on (i.e., vary with) initialization. Nonetheless, empirical evidence presented in Section 4.5 suggests that the proposed transform learning scheme is insensitive to initialization. This leads us to

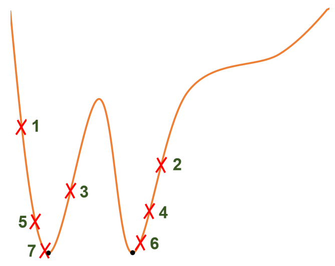


Figure 4.2: Possible behavior of the algorithm near two hypothetical local minima (marked with black dots) of the objective. The numbered iterate sequence here has two subsequences (one even numbered, and one odd numbered) that converge to the two equally good (i.e., corresponding to the same value of the objective) local minima.

conjecture that our algorithm could potentially converge to the global minimizer(s) of the learning problem in some (practical) scenarios. Fig. 4.2 provides a simple illustration of the convergence behavior of our algorithm. We also have the following corollary of Theorem 1, where 'globally convergent' refers to convergence from any initialization.

Corollary 4. *Algorithm A1 is globally convergent to the set of local minimizers of the non-convex transform learning objective $g(W, X)$.*

Note that our convergence result for the proposed non-convex learning algorithm, is free of any extra conditions or requirements. This is in clear distinction to algorithms such as IHT [102, 103] that solve non-convex problems, but require extra stringent conditions (e.g., tight conditions on restricted isometry constants of certain matrices) for their convergence results to hold. Theorem 1 also holds for any choice of the parameter λ_0 (or, equivalently λ) in (P4.0), that controls the condition number.

The optimality condition (4.16) in Theorem 1 holds true not only for local (small) perturbations in X, but also for arbitrarily large perturbations of X in a half space. For a particular accumulation point (W, X), the condition $tr\left\{(WY - X)\Delta X^T\right\} \leq 0$ in Theorem 1 defines a half-space of permissible perturbations in $\mathbb{R}^{n\times N}$. Now, even among the perturbations outside this half-space, i.e., ΔX satisfying $tr\left\{(WY - X)\Delta X^T\right\} > 0$ (and also outside the local region R2 in Theorem 1), we only need to be concerned about the perturbations that maintain the sparsity level, i.e., ΔX such that $X + \Delta X$ has sparsity $\leq s$ per column. For any other ΔX, $g(W + dW, X + \Delta X) =$

$+\infty > g(W, X)$ trivially. Now, since $X + \Delta X$ needs to have sparsity $\leq s$ per column, ΔX itself can be at most $2s$ sparse per column. Therefore, the condition $tr\left\{(WY - X)\Delta X^T\right\} > 0$ (corresponding to perturbations that could violate (4.16)) essentially corresponds to a union of low dimensional half-spaces (each corresponding to a different possible choice of support of ΔX). In other words, the set of "bad" perturbations is vanishingly small in $\mathbb{R}^{n \times N}$.

Note that Problem (P4.0) can be directly used for adaptive sparse representation (compression) of images [9, 88], in which case the convergence results here are directly applicable. (P4.0) can also be used in applications such as blind denoising [2], and blind compressed sensing [104]. The overall problem formulations [2, 104] in these applications are highly non-convex (see Section 4.5.4). However, the problems are solved using alternating optimization [2, 104], and the transform learning Problem (P4.0) arises as a sub-problem. Therefore, by using the proposed learning scheme, the transform learning step of the alternating algorithms for denoising/compressed sensing can be guaranteed (by Theorem 1) to converge [3].

4.4.2 Result for Penalized Problem

When the sparsity constraints in (P4.0) are replaced with ℓ_0 penalties in the objective with weights η_i^2 (i.e., we solve the problem corresponding to equation (4.2)), we obtain an unconstrained transform learning problem with objective $u(W, X) = \|WY - X\|_F^2 + \lambda\xi \|W\|_F^2 - \lambda \log |\det W| + \sum_{i=1}^N \eta_i^2 \|X_i\|_0$. In this case too, we have a convergence guarantee (similar to Theorem 1) for Algorithm A2 that minimizes $u(W, X)$.

Theorem 2. *Let $\left\{W^k, X^k\right\}$ denote the iterate sequence generated by Algorithm A2 with training data Y and initial (W^0, X^0). Then, the objective sequence $\left\{u(W^k, X^k)\right\}$ is monotone decreasing, and converges to a finite value, say $u^* = u^*\left(W^0, X^0\right)$. Moreover, the iterate sequence is bounded, and any specific accumulation point (W, X) of the iterate sequence is a fixed point of*

[3]Even when different columns of X are required to have different sparsity levels in (P4.0), our learning algorithm and Theorem 1 can be trivially modified to guarantee convergence.

the algorithm satisfying the following local optimality condition.

$$u(W + dW, X + \Delta X) \geq u(W, X) = u^* \tag{4.17}$$

The condition holds for sufficiently small $dW \in \mathbb{R}^{n\times n}$ satisfying $\|dW\|_F \leq \epsilon'$ for some $\epsilon' > 0$ that depends on the specific W, and all $\Delta X \in \mathbb{R}^{n\times N}$ satisfying $\|\Delta X\|_\infty < \min_i \{\eta_i/2\}$.

The proofs of Theorems 1 and 2 are provided in Appendix B.3 and B.5. Owing to Theorem 2, results analogous to Corollaries 3 and 4 apply.

Corollary 5. *Corollaries 3 and 4 apply to Algorithm A2 and the corresponding objective $u(W, X)$ as well.*

4.5 Experiments

4.5.1 Framework

In this section, we present results demonstrating the properties of our proposed transform learning Algorithm A1 for (P4.0), and its usefulness in applications. First, we illustrate the convergence behavior of our alternating learning algorithm. We consider various initializations for transform learning and investigate whether the proposed algorithm is sensitive to initializations. This study will provide some (limited) empirical understanding of local/global convergence behavior of the algorithm. Then, we compare our proposed algorithm to the NLCG-based transform learning algorithm [9] at various patch sizes, in terms of image representation quality and computational cost of learning. Finally, we briefly discuss the usefulness of the proposed scheme in image denoising.

All our implementations were coded in Matlab version R2013a. All computations were performed with an Intel Core i5 CPU at 2.5GHz and 4GB memory, employing a 64-bit Windows 7 operating system.

The data in our experiments are generated as the 2D patches of natural images. We use our transform learning Problem (P4.0) to learn adaptive sparse representations of such image patches. The means of the patches are removed and we only sparsify the mean-subtracted patches which are stacked as columns of the training matrix Y (patches reshaped as vectors) in (P4.0). The means are added back for image display. Mean removal is typically adopted in image processing applications such as compression and denoising [86, 2]. Similar to prior work [9, 2], the weight $\xi = 1$ in all our experiments.

We have previously introduced several metrics in Chapter 2 to evaluate the quality of learned transforms [9, 2]. The *normalized sparsification error* (NSE) for a transform W is defined as $\|WY - X\|_F^2 / \; \|WY\|_F^2$, where Y is the data matrix, and the columns $X_i = H_s(WY_i)$ of the matrix X denote the sparse codes. The recovery peak signal to noise ratio (*recovery PSNR*) was previously defined (in decibels (dB)) as the scaled (by the factor 20) base-10 logarithm of $255\sqrt{P} / \; \|Y - W^{-1}X\|_F$, where P is the number of image pixels and X is again the transform sparse code of data Y. The recovery PSNR serves as a simple surrogate for the performance of the learned transform in compression. Note that if the proposed approach were to be used for compression, then the W matrix too would have to be transmitted as side information.

4.5.2 Convergence Behavior

Here, we study the convergence behavior of the proposed transform learning Algorithm A1. We extract the 8×8 ($n = 64$) non-overlapping (mean-subtracted) patches of the 512×512 image Barbara [17]. Problem (P4.0) is solved to learn a square transform W that is adapted to this data. The data matrix Y in this case has $N = 4096$ training signals (patches represented as vectors). The parameters are set as $s = 11$, $\lambda_0 = 3.1 \times 10^{-3}$. The choice of λ_0 here ensures well-conditioning of the learned transform. Badly conditioned transforms degrade performance in applications [9, 2]. Hence, we focus our investigation here only on the well-conditioned scenario.

We study the convergence behavior of Algorithm A1 for various initializations of W. Once W is initialized, the algorithm iterates over the sparse coding and transform update steps (this corresponds to a different ordering of the steps in Fig. 4.1). We consider four different initializations (initial transforms) for the algorithm. The first is the 64×64 2D DCT matrix (obtained as the Kronecker product of two 8×8 1D DCT matrices). The second initialization is the Karhunen-Loève Transform (KLT) (i.e., the inverse of

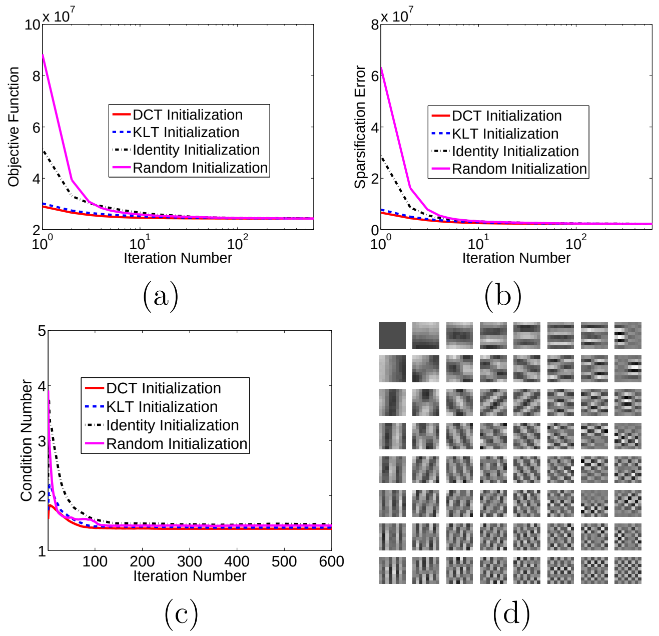


Figure 4.3: Effect of different Initializations: (a) Objective function, (b) Sparsification error, (c) Condition number, (d) Rows of the learned transform shown as patches for the case of DCT initialization.

PCA), obtained here by inverting/transposing the left singular matrix of Y [4]. The third and fourth initializations are the identity matrix, and a random matrix with i.i.d. Gaussian entries (zero mean and standard deviation 0.2), respectively.

Figure 4.3 shows the progress of the algorithm over iterations for the various initializations of W. The objective function (Fig. 4.3(a)), sparsification error (Fig. 4.3(b)), and condition number (Fig. 4.3(c)), all converge quickly for our algorithm. The sparsification error decreases over the iterations, as required. Importantly, the final values of the objective (similarly, the sparsification error, and condition number) are nearly identical for all the initializations. This indicates that our learning algorithm is reasonably robust, or insensitive to initialization. Good initializations for W such as the DCT and KLT lead to faster convergence of learning. The learned transforms also have identical Frobenius norms (5.14) for all the initializations.

Figure 4.3(d) shows the (well-conditioned) transform learned with the DCT

[4]We did not remove the means of the rows of Y here. However, we obtain almost identical plots in Fig. 4.3, when the learning algorithm is instead initialized with the KLT computed on (row) mean centered data Y.

initialization. Each row of the learned W is displayed as an 8×8 patch, called the transform atom. The atoms here exhibit frequency and texture-like structures that sparsify the patches of Barbara. Similar to our prior work [9], we observed that the transforms learned with different initializations, although essentially equivalent in the sense that they produce similar sparsification errors and are similarly scaled and conditioned, appear somewhat different (i.e., they are not related by only row permutations and sign changes). The transforms learned with different initializations in Fig. 4.3 also provide similar recovery PSNRs (that differ by hundredths of a dB) for the Barbara image.

4.5.3 Image Representation

For the second experiment, we learn sparsifying transforms from the $\sqrt{n} \times \sqrt{n}$ (zero mean) non-overlapping patches of the image Barbara at various patch sizes n. We study the image representation performance of the proposed algorithm involving closed-form solutions for Problem (P4.0). We compare the performance of our algorithm to the NLCG-based algorithm [9] that solves a version (without the absolute value within the log-determinant) of (P4.0), and the fixed 2D DCT. The DCT is a popular analytical transform that has been extensively used in compression standards such as JPEG. We set $s = 0.17 \times n$ (rounded to nearest integer), and λ_0 is fixed to the same value as in Section 4.5.2 for simplicity. The NLCG-based algorithm is executed with 128 NLCG iterations for each transform update step, and a fixed step size of 10^{-8} [9].

Figure 4.4 plots the normalized sparsification error (Fig. 4.4(a)) and recovery PSNR (Fig. 4.4(b)) metrics for the learned transforms, and for the patch-based 2D DCT, as a function of patch size. The run times of the various transform learning schemes (Fig. 4.4(c)) are also plotted.

The learned transforms provide better sparsification and recovery than the analytical DCT at all patch sizes. The gap in performance between the adapted transforms and the fixed DCT also increases with patch size (cf. [2] for a similar result and the reasoning). The learned transforms in our experiments are all well-conditioned (condition numbers $\approx 1.2 - 1.6$). Note that the performance gap between the adapted transforms and the DCT can

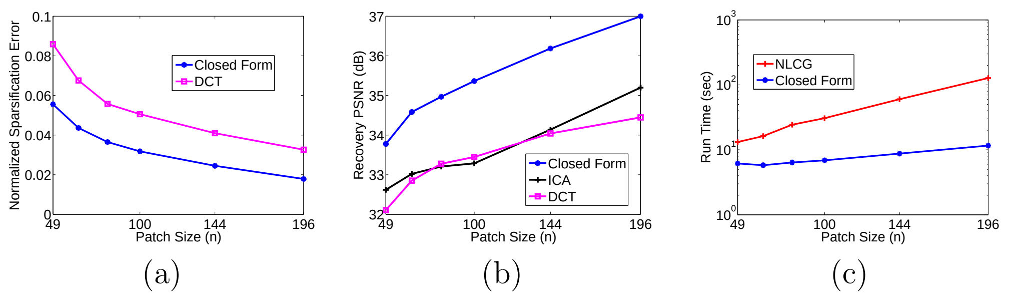


Figure 4.4: Comparison of NLCG-based transform learning [9], Closed Form transform learning via (P4.0), DCT, and ICA [10] for different patch sizes: (a) Normalized sparsification error, (b) Recovery PSNR, (c) Run time of transform learning. The plots for the NLCG and Closed Form methods overlap in (a) and (b). Therefore, we only show the plots for the Closed Form method there.

be amplified further at each patch size, by optimal choice of λ_0 (or, optimal choice of condition number [5]).

The performance (normalized sparsification error and recovery PSNR) of the NLCG-based algorithm [9] is identical to that of the proposed Algorithm A1 for (P4.0) involving closed-form solutions. However, the latter is much faster (by 2-11 times) than the NLCG-based algorithm. The actual speedups depend in general, on how J (the number of NLCG iterations) scales with respect to N/n.

In yet another comparison, we show in Fig. 4.4(b), the recovery PSNRs obtained by employing Independent Component Analysis (ICA – a method for blind source separation) [105, 106, 107, 10, 108]. Similar to prior work on ICA-based image representation [109], we learn an ICA model A (a basis here) using the FastICA algorithm [10, 110], to represent the training signals as $Y = AZ$, where the rows of Z correspond to independent sources. Note that the ICA model enforces different properties (e.g., independence) than the transform model. Once the ICA model is learned (using default settings in the author's MATLAB implementation [110]), the training signals are sparse coded in the learned ICA model A [109] using the orthogonal matching pursuit algorithm [34], and the recovery PSNR (defined as in Section 4.5.1,

[5]The recovery PSNR depends on the trade-off between the sparsification error and condition number [9, 2]. For natural images, the recovery PSNR using the learned transform is typically better at λ values corresponding to intermediate conditioning or well-conditioning, rather than unit conditioning, since unit conditioning is too restrictive [9].

but with $W^{-1}X$ replaced by $A\hat{Z}$, where $\hat{Z}$ is the sparse code in the ICA basis) is computed. We found that the $A^{\dagger}$ obtained using the FastICA algorithm provides poor normalized sparsification errors (i.e., it is a bad transform model). Therefore, we only show the recovery PSNRs for ICA. As seen in Fig. 4.4(b), the proposed transform learning algorithm provides better recovery PSNRs than the ICA approach. This illustrates the superiority of the transform model for sparse representation (compression) of images compared to ICA. While we used the FastICA algorithm in Fig. 4.4(b), we have also observed similar performance for alternative (but slower) ICA methods [108, 111].

Finally, in comparison to synthesis dictionary learning, we have observed that algorithms such as K-SVD [17] perform slightly better than the transform learning Algorithm A1 for the task of image representation. However, the learning and application of synthesis dictionaries also imposes a heavy computational burden (cf. [9] for a comparison of the run times of synthesis K-SVD and NLCG-based transform learning). Indeed, an important advantage of our transform-based scheme for a compression application (similar to classical approaches involving the DCT or Wavelets), is that the transform can be applied as well as learned very cheaply.

While we adapted the transform to a specific image (i.e., image-specific transform) in Fig. 4.4, a transform adapted to a variety of images (global transform) also performs well in test images [88]. Both global and image-specific transforms may hold promise for compression.

4.5.4 Image Denoising

The goal of denoising is to recover an estimate of an image $x \in \mathbb{R}^P$ (2D image represented as a vector) from its corrupted measurement $y = x + h$, where h is the noise. We work with h whose entries are i.i.d. Gaussian with zero mean and variance σ^2. We have previously presented a formulation [2] for patch-based image denoising using adaptive transforms as follows.

$$\min_{W,\{x_i\},\{\alpha_i\}} \sum_{i=1}^{N} \left\{ \|Wx_i - \alpha_i\|_2^2 + \lambda_i v(W) + \tau \|R_i\, y - x_i\|_2^2 \right\}$$
$$s.t. \quad \|\alpha_i\|_0 \leq s_i \;\; \forall\, i \qquad \text{(P4.1)}$$

Here, $R_i \in \mathbb{R}^{n \times P}$ extracts the i$^{\text{th}}$ patch (N overlapping patches assumed) of the image y as a vector $R_i y$. Vector $x_i \in \mathbb{R}^n$ denotes a denoised version of $R_i y$, and $\alpha_i \in \mathbb{R}^n$ is a sparse representation of x_i in a transform W, with an a priori unknown sparsity s_i. The weight $\tau \propto 1/\sigma$ [1, 2], and λ_i is set based on the given noisy data $R_i y$ as $\lambda_0 \|R_i y\|_2^2$. The net weighting on $v(W)$ in (P4.1) is then $\lambda = \sum_i \lambda_i$.

We have previously proposed a simple *two-step* iterative algorithm to solve (P4.1) [2] (see also Chapter 3), that also estimates the unknown s_i. The algorithm iterates over a transform learning step and a variable sparsity update step (cf. [2] for a full description of these steps). We use the proposed alternating transform learning Algorithm A1 (involving closed-form updates) in the transform learning step. Once the denoised patches x_i are found, the denoised image x is obtained by averaging the x_i's at their respective locations in the image [2].

We now present brief results for our denoising framework employing the proposed efficient closed-form solutions in transform learning. We work with the images Barbara, Cameraman, Couple [6], and Brain (same as the one in Fig. 1 of [5]), and simulate i.i.d. Gaussian noise at 5 different noise levels (σ = 5, 10, 15, 20, 100) for each of the images. We compare the denoising results and run times obtained by our proposed algorithm with those obtained by the adaptive overcomplete synthesis K-SVD denoising scheme [1]. The Matlab implementation of K-SVD denoising [1] available from Michael Elad's website [86] was used in our comparisons, and we used the built-in parameter settings of that implementation.

We use 11×11 maximally overlapping image patches for our transform-based scheme. The resulting 121×121 square transform [7] has about the same number of free parameters as the 64×256 overcomplete K-SVD dictionary [1, 86]. The settings for the various parameters (not optimized) in our transform-based denoising scheme are listed in Table 4.1. At $\sigma = 100$, we set the number of iterations of the two-step denoising algorithm [2] to $M' = 5$ (lower than the value in Table 4.1), which also works well, and provides slightly smaller run times in denoising.

[6]These three well-known images have been used in our previous work [2].

[7]We have previously shown reasonable denoising performance for adapted (using NLCG-based transform learning [9]) 64×64 transforms [2]. The denoising performance usually improves when the transform size is increased, but with some degradation in run time.

Parameter	Value	Parameter	Value
n	121	N'	32000
λ_0	0.031	τ	$0.01/\sigma$
C	1.04	s	12
M'	11	M	12

Table 4.1: The parameter settings for our algorithm: n - number of pixels in a patch, λ_0 - weight in (P4.1), C - sets threshold that determines sparsity levels in the variable sparsity update step [2], M' - number of iterations of the two-step denoising algorithm [2], N' - training size for the transform learning step (the training patches are chosen uniformly at random from all patches in each denoising iteration) [2], M - number of iterations in transform learning step, τ - weight in (P4.1), s - initial sparsity level for patches [2].

Table 4.2 lists the denoising PSNRs obtained by our transform-based scheme, along with the PSNRs obtained by K-SVD. The transform-based scheme provides better PSNRs than K-SVD for all the images and noise levels considered. The average PSNR improvement (averaged over all rows of Table 4.2) provided by the transform-based scheme over K-SVD is 0.18 dB. When the NLCG-based transform learning [9] is used in our denoising algorithm, the denoising PSNRs obtained are very similar to the ones shown in Table 4.2 for the algorithm involving closed-form updates. However, the latter scheme is faster.

We also show the average speedups provided by our transform-based denoising scheme [8] over K-SVD denoising in Table 4.3. For each image and noise level, the ratio of the run times of K-SVD denoising and transform denoising (involving closed-form updates) is first computed, and these speedups are averaged over the four images at each noise level. The transform-based scheme is about 10x faster than K-SVD denoising at lower noise levels. Even at very high noise ($\sigma = 100$), the transform-based scheme is still computationally cheaper than the K-SVD method.

We observe that the speedup of the transform-based scheme over K-SVD denoising decreases as σ increases in Table 4.3. This is mainly because the computational cost of the transform-based scheme is dominated by matrix-

[8]Our MATLAB implementation is not currently optimized for efficiency. Therefore, the speedups here are computed by comparing our unoptimized MATLAB implementation (for transform-based denoising) to the corresponding MATLAB implementation [86] of K-SVD denoising.

Image	σ	Noisy PSNR	K-SVD	Transform
Barbara	5	34.15	38.09	38.28
	10	28.14	34.42	34.55
	15	24.59	32.34	32.39
	20	22.13	30.82	30.90
	100	8.11	21.86	22.42
Cameraman	5	34.12	37.82	37.98
	10	28.14	33.72	33.87
	15	24.60	31.50	31.65
	20	22.10	29.83	29.96
	100	8.14	21.75	22.01
Brain	5	34.14	42.14	42.74
	10	28.12	38.54	38.78
	15	24.62	36.27	36.43
	20	22.09	34.70	34.71
	100	8.13	24.73	24.83
Couple	5	34.16	37.29	37.35
	10	28.11	33.48	33.67
	15	24.59	31.44	31.60
	20	22.11	30.01	30.17
	100	8.13	22.58	22.60

Table 4.2: PSNR values in decibels for denoising with adaptive transforms, along with the corresponding values for 64×256 overcomplete K-SVD [1]. The PSNR values of the noisy images (denoted as Noisy PSNR) are also shown.

vector multiplications (see [2] and Section 4.3.3), and is invariant to the sparsity level s. On the other hand, the cost of the K-SVD denoising method is dominated by synthesis sparse coding, which becomes cheaper as the sparsity level decreases. Since sparsity levels in K-SVD denoising are set according to an error threshold criterion (and the error threshold $\propto \sigma^2$) [1, 86], they decrease with increasing noise in the K-SVD scheme. For these reasons, the speedup of the transform method over K-SVD is lower at higher noise levels in Table 4.3.

We would like to point out that the actual value of the speedup over K-SVD also depends on the patch size used (by each method). For example, for larger images, a larger patch size would be used to capture image information better. The sparsity level in the synthesis model typically scales as a fraction of the patch size (i.e., $s \propto n$). Therefore, the actual speedup of transform-based denoising over K-SVD at a particular noise level would increase with

σ	5	10	15	20	100
Average Speedup	9.82	8.26	4.94	3.45	2.16

Table 4.3: The denoising speedups provided by our transform-based scheme (involving closed-form solutions) over K-SVD [1]. The speedups are averaged over the four images at each noise level.

increasing patch (and image) size – an effect that is not fully explored here due to limitations of space.

Thus, here, we have shown the promise of the transform-based denoising scheme (involving closed-form updates in learning) over overcomplete K-SVD denoising. Adaptive transforms provide better denoising, and are faster. The denoising PSNRs shown for adaptive transforms in Table 4.2 become even better at larger transform sizes, or by optimal choice of parameters [9]. We plan to combine transform learning with the state-of-the-art denoising scheme BM3D [7] in the near future. Since the BM3D algorithm involves some sparsifying transformations, we conjecture that adapting such transforms could improve the performance of the algorithm.

4.6 Conclusions

In this chapter, we investigated the problem formulations for learning orthonormal as well as well-conditioned square sparsifying transforms. The proposed iterative algorithms involve efficient optimal updates for the sparse coding and transform update steps. Importantly, we provided strong convergence guarantees for our transform learning algorithms. The learned transforms provide better representations than analytical ones such as the DCT for images. Moreover, our algorithm is faster than previous ones (in Chapter 2) involving iterative NLCG in the transform update step. In the application of image denoising, our algorithms provide comparable or better performance over the synthesis K-SVD, while being faster.

[9]The parameter settings in Table 4.1 (used in all our experiments for simplicity) can be optimized for each noise level, similar to [2].

CHAPTER 5

BLIND COMPRESSED SENSING USING SPARSIFYING TRANSFORMS WITH CONVERGENCE GUARANTEES AND APPLICATION TO MRI

5.1 Introduction

Sparsity-based techniques have become extremely popular in inverse problems in image processing and imaging in recent years. These techniques typically exploit the sparsity of images or image patches in a transform domain or dictionary to restore/reconstruct images from measurements. In this chapter, we investigate the subject of blind compressed sensing, which aims to reconstruct images in the scenario when a good sparse model for the image is unknown a priori. In particular, we will work with the sparsifying transform model [9] that has been shown to be useful in various imaging applications. In the following, we briefly review the topics of compressed sensing, and blind compressed sensing. We then list the contributions of this work.

5.1.1 Compressed Sensing

In the context of imaging, the recent theory of Compressed Sensing (CS) [30, 31, 29] (see also [112, 113, 114, 115, 116, 117, 118, 119] for the earliest versions of CS for Fourier-sparse signals and for Fourier imaging) enables accurate recovery of images from significantly fewer measurements than the number of unknowns. In order to do so, it requires that the underlying image be sufficiently sparse in some transform domain or dictionary, and that the measurement acquisition procedure is incoherent, in an appropriate sense, with the transform. However, the image reconstruction procedure for compressed sensing is typically computationally expensive and non-linear.

The image reconstruction problem in compressed sensing is typically formulated as

$$\min_{x} \|\Psi x\|_0 \quad s.t.\ Ax = y \tag{5.1}$$

Here, $x \in \mathbb{C}^p$ is a vectorized representation of the image (obtained by stacking the image columns on top of each other) to be reconstructed, and $y \in \mathbb{C}^m$ denotes the measurements. The operator $A \in \mathbb{C}^{m\times p}$, with $m \ll p$ is known as the sensing matrix, or measurement matrix. The matrix $\Psi \in\in \mathbb{C}^{t\times p}$ is a sparsifying transform (typically chosen as orthonormal). The aim of Problem (5.1) is to find the image satisfying the measurement equation $Ax = y$, that is the sparsest possible in the Ψ-transform domain. Since, in CS, the measurement equation $Ax = y$ represents an underdetermined system of equations, an additional model (such as the sparsity model above) is needed to estimate the true underlying image.

When Ψ is orthonormal, Problem (5.1) can be rewritten as

$$\min_{z} \|z\|_0 \quad s.t.\ A\Psi^H z = y \tag{5.2}$$

where we used the substitution $\Psi x = z$, and $(\cdot)^H$ denotes the matrix Hermitian (conjugate transpose) operation. Similar to the synthesis sparse coding problem, Problem (5.2) too is NP-hard. Often the l_0 "norm" in (5.1) is replaced with its convex relaxation, the l_1 norm [40], and the following convex problem is solved to reconstruct the image, when the CS measurements are noisy [3, 120].

$$\min_{x} \|Ax - y\|_2^2 + \lambda \|\Psi x\|_1 \tag{5.3}$$

In Problem (5.3), the ℓ_2 penalty for the measurement fidelity term can also be replaced with alternative penalties such as a weighted ℓ_2 penalty, depending on the physics of the measurement process and the statistics of the measurement noise.

Recently, CS theory has been applied to imaging techniques such as magnetic resonance imaging (MRI) [3, 78, 121, 122, 123, 124, 125], computed tomography (CT) [126, 127, 128], and Positron emission tomography (PET) imaging [129, 130], demonstrating high quality reconstructions from a reduced set of measurements. Such compressive measurements are highly advantageous in these applications. For example, they help reduce the radiation dosage in CT, and reduce scan times and improve clinical throughput in MRI. Well-known inverse problems in image processing such as inpainting (where an image is reconstructed from a subset of measured pixels) can also be viewed as compressed sensing problems.

5.1.2 Blind Compressed Sensing

While conventional compressed sensing techniques utilize fixed analytical sparsifying transforms such as wavelets [12], finite differences, and contourlets [46], to reconstruct images, in this work, we instead focus on the idea of blind compressed sensing (BCS) [5, 64, 131, 132, 133], where the underlying sparse model is assumed unknown a priori. The goal of blind compressed sensing is to simultaneously reconstruct the underlying image(s) as well as the dictionary or transform from highly undersampled measurements. Thus, BCS enables the sparse model to be adaptive to the specific data under consideration. Recent research has shown that such data-driven adaptation of dictionaries or transforms is advantageous in many applications [1, 56, 58, 59, 61, 5, 2, 88, 96, 134]. While the adaptation of synthesis dictionaries [15, 16, 17, 18, 19, 20] has been extensively studied, recent work has shown advantages in terms of computation and application-specific performance, for the adaptation of transform models [9, 96, 134].

In a prior work on BCS [5], we successfully demonstrated the usefulness of dictionary-based blind compressed sensing for MRI, even in the case when the undersampled measurements corresponding to only a single image are provided. In the latter case, the overlapping patches of the underlying image are assumed to be sparse in a dictionary, and the (unknown) patch-based dictionary, that is typically much smaller in size than the image, is learned directly from the compressive measurements.

BCS techniques have been demonstrated to provide much better image reconstruction quality compared to compressed sensing methods that utilize a fixed sparsifying transform or dictionary [5, 132, 133]. This is not surprising since BCS methods allow for data-specific adaptation, and data-specific dictionaries typically sparsify the underlying images much better than analytical ones.

5.1.3 Contributions

5.1.3.1 Highlights

The BCS framework assumes a specific sparse model for the underlying image(s) or image patches. While prior work on BCS primarily focused on the synthesis model, in this work, we instead focus on the sparsifying transform model. We propose novel problem formulations for BCS involving well-conditioned or orthonormal adaptive *square* sparsifying transforms. Our framework simultaneously adapts the sparsifying transform and reconstructs the underlying image(s) from highly undersampled measurements. We propose efficient block coordinate descent-type algorithms for transform-based BCS. Importantly, we establish that our iterative algorithms are *globally convergent* (i.e., they converge from any initialization) to the set of critical points of the proposed highly non-convex BCS problems. These critical points are guaranteed to be *at least* partial global or local minimizers. Such convergence guarantees have not been established for prior *blind* compressed sensing methods.

Note that although we focus on compressed sensing in the discussions and experiments of this chapter, the formulations and algorithms proposed by us can also handle the case when the measurement/sensing matrix A is square (e.g., in signal denoising), or even overcomplete (e.g., deconvolution).

5.1.3.2 Magnetic Resonance Imaging Application

MRI is a non-invasive and non-ionizing imaging technique that offers a variety of contrast mechanisms, and enables excellent visualization of anatomical structures and physiological functions. However, the data in MRI, which are samples in k-space of the spatial Fourier transform of the object, are acquired sequentially in time. Hence, a major drawback of MRI, that affects both clinical throughput and image quality especially in dynamic imaging applications, is that it is a relatively slow imaging technique. Although there have been advances in scanner hardware [135] and pulse sequences, the rate at which MR data are acquired is limited by MR physics and physiological constraints on RF energy deposition. Compressed sensing MRI (either blind, or with known sparse model) has become quite popular in recent years, and it alleviates some of the aforementioned problems by enabling accurate reconstruction of MR images from highly undersampled measurements.

In this chapter, we illustrate the usefulness of the proposed transform-based BCS schemes for magnetic resonance image reconstruction from highly undersampled k-space data. We show that our adaptive transform-based BCS provides better image reconstruction quality compared to prior methods that involve fixed image-based, or patch-based sparsifying transforms. Importantly, transform-based BCS is shown to be 10x faster than synthesis dictionary-based BCS [5] for reconstructing 2D MRI data. The speedup is expected to be much higher when considering 3D or 4D MR data. These advantages make the proposed scheme more amenable for adoption for clinical use in MRI.

5.1.4 Organization

The rest of this chapter is organized as follows. Section 5.2 describes our transform learning-based blind compressed sensing formulations and their properties. In Section 5.3, we derive efficient block coordinate descent algorithms for solving the BCS Problem, and discuss the algorithms' computational cost. In Section 5.4, we present novel convergence guarantees for our algorithms. The proof of convergence is provided in Appendix C. Section 5.5 presents experimental results demonstrating the convergence behavior, performance, and computational efficiency of the proposed scheme for the MRI application. In Section 5.6, we conclude.

5.2 Problem Formulations

5.2.1 Synthesis dictionary-based Blind Compressed Sensing

CS image reconstructions employing fixed, non-adaptive sparsifying transforms typically suffer from many artifacts at high undersampling factors [5]. Blind compressed sensing allows for the sparse model to be directly adapted to the object(s) being imaged. For example, the overlapping patches of the underlying image may be assumed to be sparse in a certain model. Hence, in prior work, we proposed a synthesis-dictionary based BCS formulation as follows

$$\text{(P5.0)} \quad \min_{x,D,B} \sum_{j=1}^{N} \|R_j x - Db_j\|_2^2 + \nu \|Ax - y\|_2^2$$
$$s.t. \;\; \|d_k\|_2 = 1 \, \forall \, k, \;\; \|b_j\|_0 \leq s \;\; \forall \, j.$$

Here, $\nu > 0$ is a weight for the measurement fidelity term ($\|Ax - y\|_2^2$), and $R_j \in \mathbb{C}^{n \times p}$ represents the operator that extracts a $\sqrt{n} \times \sqrt{n}$ 2D patch as a vector from the image x as $R_j x \in \mathbb{C}^n$. A total of N overlapping 2D patches are used. The synthesis model allows each patch $R_j x$ to be approximated by a linear combination Db_j of a small number of columns from a dictionary $D \in \mathbb{C}^{n \times K}$, where $b_j \in \mathbb{C}^K$ is sparse. The columns of the learned dictionary (represented by $d_k, 1 \leq k \leq K$) in (P5.0) are additionally constrained to be of unit norm in order to avoid the scaling ambiguity [48]. The dictionary, and the image patch, are assumed to be much smaller than the size of the image ($n, K \ll p$) in (P5.0). Problem (P5.0) thus enforces all the N (a typically large number) overlapping image patches to be sparse in some dictionary D, which can be considered as a strong prior on the underlying image.

We use $B \in \mathbb{C}^{n \times N}$ to denote the matrix that has the sparse codes of the patches b_j as its columns. Each sparse code is permitted a maximum sparsity level of $s \ll n$ in (P5.0). Although a single sparsity level is used for all patches in (P5.0) for simplicity, in practice, different sparsity levels may be allowed for different patches (for example, by setting an appropriate error threshold in the sparse coding step of optimization algorithms [5]). For the case of MRI, the sensing matrix A in (P5.0) is $F_u \in \mathbb{C}^{m \times p}$, the undersampled Fourier encoding matrix [5]. The weight ν in (P5.0) is set depending on the measurement noise level (σ) as $\nu = \frac{\lambda}{\sigma}$, where λ is a positive constant [5]. In practice, an estimate of the noise level or the observed noise level can be used to determine ν.

Problem (P5.0) learns a patch-based synthesis sparsifying dictionary ($n, K \ll p$), and reconstructs the image simultaneously from highly undersampled measurements. As discussed before, we have previously shown significantly superior image reconstructions for MRI using (P5.0), as compared to non-adaptive compressed sensing schemes that solve Problem (5.3). However, the BCS Problem (P5.0) is both non-convex and NP-hard. Approximate iterative algorithms for (P5.0) (e.g., the DLMRI algorithm [5]) typically solve the synthesis sparse coding problem repeatedly, which makes them computationally expensive. Moreover, no convergence guarantees exist for the algorithms that solve (P5.0).

5.2.2 Sparsifying Transform-based Blind Compressed Sensing

In order to overcome some of the aforementioned drawbacks of synthesis dictionary-based BCS, we propose using the sparsifying transform model in this work. Sparsifying transform learning has been shown to be effective and efficient in applications, while also enjoying good convergence guarantees [96]. Our problem formulation for BCS employing adaptive sparsifying transforms is as follows:

$$\text{(P5.1)} \quad \min_{x,W,B} \sum_{j=1}^{N} \|WR_jx - b_j\|_2^2 + \nu \, \|Ax - y\|_2^2 + \lambda \, Q(W)$$
$$s.t. \ \sum_{j=1}^{N} \|b_j\|_0 \leq s, \ \ \|x\|_2 \leq C.$$

Here, $W \in \mathbb{C}^{n\times n}$ denotes a square sparsifying transform for the patches of the underlying image. The penalty $\|WR_jx - b_j\|_2^2$ denotes the sparsification error (transform domain residual) for the j$^{\text{th}}$ patch, with b_j denoting the transform sparse code. Notice that the sparsity constraint is enforced on all the overlapping patches, taken together. This is a way of enabling variable sparsity levels for each specific patch. The constraint $\|x\|_2 \leq C$ with $C > 0$ in (P5.1), is to enforce any prior knowledge on the signal energy (or, range). For example, if the pixels of the underlying image take intensity values in the range $0 - 255$, then $C = 255\sqrt{p}$ is an appropriate bound. The function $Q(W) : \mathbb{C}^{n\times n} \mapsto \mathbb{R}$ in Problem (P5.1) denotes a regularizer for the transform, and the weight $\lambda > 0$. Notice that without an additional regularizer, $W = 0$ is a trivial sparsifier for any patch, and therefore, $W = 0$, $b_j = 0 \ \forall j$, $x = A^{\dagger}y$ (assuming this x satisfies $\|x\|_2 \leq C$) with $(\cdot)^{\dagger}$ denoting the pseudo-inverse, would trivially minimize the objective (without the regularizer)in Problem (P5.1).

Similar to prior work on transform learning [9, 6], we set $Q(W) \triangleq -\log\,|\det W| + 0.5\,\|W\|_F^2$ as the regularizer in the objective to prevent trivial solutions. The $-\log\,|\det W|$ penalty eliminates degenerate solutions such as those with repeated rows. The $\|W\|_F^2$ penalty helps remove a 'scale ambiguity' in the solution [9], which occurs when the optimal solution satisfies an exactly sparse representation, i.e., the optimal (x, W, B) in (P5.1) is such that $WR_jx = b_j \; \forall \, j$, and $\sum_{j=1}^N \|b_j\|_0 \leq s$. In this case, if the $\|W\|_F^2$ penalty is absent in (P5.1), the optimal (W, B) can be scaled by $\beta \in \mathbb{C}$, with $|\beta| \to \infty$, which causes the objective to decrease unbounded.

The $-\log|\det W|$ and $0.5\,\|W\|_F^2$ penalties together also additionally help control the condition number $\kappa(W)$ and scaling of the learned transform. If we were to minimize only the $Q(W)$ regularizer in Problem (P5.1) with respect to W, then the minimum is achieved with a W that is unit conditioned, and with spectral norm (scaling) of 1 [9], i.e., a *unitary* or *orthonormal* transform W. Thus, similar to Corollary 2 in [9], it is easy to show that as $\lambda \to \infty$ in Problem (P5.1), the optimal sparsifying transform(s) tends to a unitary one. In practice, transforms learned via (P5.1) are typically almost unitary even for finite but large λ. Adaptive well-conditioned transforms (small $\kappa(W) \neq 1$) have been previously shown to perform better than adaptive (strictly) orthonormal ones in some scenarios in image representation, or image denoising [9, 6].

In this chapter, we set $\lambda = \lambda_0 N$ in (P5.1), where $\lambda_0 > 0$ is a constant. This setting allows λ to scale with the size of the data (i.e., total number of patches). In practice, the weight λ_0 needs to be set according to the expected range (in intensity values) of the underlying image, as well as depending on the desired condition number of the learned transform. The weight ν in (P5.1) is set similarly as in (P5.0).

When a unitary sparsifying transform is preferred, the $Q(W)$ regularizer in (P5.1) could instead be replaced by the constraint $W^HW = I$, where I denotes the identity matrix, yielding the following formulation:

$$\text{(P5.2)} \quad \min_{x,W,B} \sum_{j=1}^{N} \|WR_jx - b_j\|_2^2 + \nu \|Ax - y\|_2^2$$
$$s.t. \;\; W^HW = I, \; \sum_{j=1}^{N} \|b_j\|_0 \leq s, \;\; \|x\|_2 \leq C.$$

The unitary sparsifying transform case is special, in that Problem (P5.2) is also a unitary synthesis dictionary-based blind compressed sensing problem, with W^H denoting the synthesis dictionary. This follows from the identity

$\|WR_jx - b_j\|_2 = \|R_jx - W^Hb_j\|_2$, for unitary W.

We have recently explored certain structures or properties for adaptive sparsifying transforms, such as double sparsity [2], or union of transforms [134]. Such structured sparsifying transforms may be useful, or provide efficiency in imaging applications. Problem (P5.1) can be appropriately modified to accommodate different transform properties. However, a detailed investigation of structured transform-based blind compressed sensing is beyond the scope of this work, and we leave it for future work.

The following simple proposition considers an "error-free" scenario and establishes the global identifiability of the underlying image and sparse model in BCS via solving the proposed Problems (P5.1) or (P5.2).

Proposition 4. *Let $x \in \mathbb{C}^p$ denote an image with $\|x\|_2 \leq C$, and let $y = Ax$ be a set of measurements obtained for the image via a sensing matrix $A \in \mathbb{C}^{m \times p}$. Suppose that $W \in \mathbb{C}^{n \times n}$ is a unitary transform model that exactly sparsifies all the patches of x as $\sum_{j=1}^{N} \|WR_jx\|_0 \leq s$. Further, let B denote the matrix that has WR_jx as its columns. Then, (x, W, B) is a global minimizer of both Problems (P5.1) and (P5.2), i.e., it is identifiable by solving the problems.*

Proof. : For the given (x, W, B), the terms $\sum_{j=1}^{N} \|WR_jx - b_j\|_2^2$ and $\|Ax - y\|_2^2$ in (P5.1) and (P5.2) each attain their minimum possible value (lower bound) of zero. Since W is unitary, the penalty $Q(W)$ in (P5.1) is also minimized by the given W. Notice that the constraints in both (P5.1) and (P5.2) are satisfied for the given (x, W, B). Therefore, this triplet is feasible for both problems and achieves the minimum possible value of the objective in both cases. Thus, it is a global minimizer of both (P5.1) and (P5.2). □

Thus, when "error-free" measurements are provided, and the patches of the underlying image are *exactly* sparse (as defined by the constraint in (P5.1)) in some unitary transform, Proposition 4 guarantees that the image as well as the model are jointly identifiable by solving (i.e., global minimizers of) (P5.1).

An interesting topic, which we do not fully pursue here due to lack of space, pertains to the condition(s) under which the underlying image in Proposition 4 is the unique minimizer of the proposed BCS problems. The proposed problems however do admit an equivalence class of solutions/minimizers with respect to the transform W and the set of sparse codes $\{b_j\}_{j=1}^N$ [1]. Given a particular minimizer (x, W, B) of (P5.1) or (P5.2), we have that $(x, \Theta W, \Theta B)$ is another equivalent minimizer for all *sparsity-preserving unitary matrices* Θ, i.e., Θ such that $\Theta^H \Theta = I$ and $\sum_j \|\Theta b_j\|_0 \leq s$. For example, Θ can be a row permutation matrix, or a diagonal ± 1 sign matrix.

While Problem (P5.1) works with a sparsity constraint, an alternative version of Problem (P5.1) is obtained by replacing the ℓ_0 sparsity constraint with an ℓ_0 penalty in the objective, in which case we have the following optimization problem:

$$\text{(P5.3)} \quad \min_{x,W,B} \sum_{j=1}^{N} \|WR_j x - b_j\|_2^2 + \nu \|Ax - y\|_2^2 + \lambda\, Q(W) + \eta^2 \sum_{j=1}^{N} \|b_j\|_0$$
$$s.t. \;\; \|x\|_2 \leq C.$$

where $\eta > 0$ denotes the weight for the sparsity penalty. A version of Problem (P5.3) (without the ℓ_2 constraint) has been used very recently in adaptive tomographic reconstruction [97, 98]. However, it is interesting to note that in the absence of the $\|x\|_2 \leq C$ condition, the objective in (P5.3) is actually non-coercive as follows. Consider $W = I$ and $x_\beta = x_0 + \beta z$, where x_0 is a particular solution to $y = Ax$, $\beta \in \mathbb{R}$, and $z \in \mathcal{N}(A)$ with $\mathcal{N}(A)$ denoting the null space of A. For this setting, as $\beta \to \infty$ with the j^{th} sparse code in (P5.3) set to $WR_j x_\beta$, it is obvious that the objective in (P5.3) always remains finite, thereby making it non-coercive. The energy constraint on x prevents this behavior. While a single weight η is used for the sparsity penalties in (P5.3), one could also use different weights η_j for the sparsity penalties in (P5.3) corresponding to different patches, if such weights are known, or estimated.

Just as Problem (P5.3) is an alternative to Problem (P5.1), we can also obtain a corresponding alternative version (denoted as (P5.4)) of Problem (P5.2) by replacing the sparsity constraint with a penalty. Although, in the rest of this chapter, we consider Problems (P5.1)-(P5.3), the proposed algorithms and convergence results in this chapter easily extend to the case

[1]In the remainder of this chapter, when certain indexed variables are enclosed within braces, it means that we are considering the set of variables over the range of all the indices.

of (P5.4).

5.2.3 Other Extensions

While the proposed sparsifying transform-based BCS problem formulations are for the (extreme) scenario when the CS measurements corresponding to a single 2D image are provided, these formulations can be easily extended to other scenarios too. For example, when multiple 2D images (or frames, or slices) have to be jointly reconstructed using a single adaptive 2D (spatial) sparsifying transform, then the objectives in Problems (P5.1)-(P5.3) for this case are the summation of the corresponding objective functions for each image. Sparsity constraints can also be enforced on (the set of patches of) each 2D image. In applications such as dynamic MRI (or for example, compressive video), the proposed formulations can be extended by considering adaptive spatiotemporal sparsifying transforms of 3D patches (cf. [133] that extends Problem (P5.0) in such a way to compressed sensing dynamic MRI). Similar extensions are also possible for higher-dimensional applications such as 4D imaging.

5.3 Algorithm and Properties

5.3.1 Algorithm

In the previous chapters, we have proposed alternating minimization algorithms in the context of sparsifying transform learning. Here, we propose block coordinate descent-type algorithms to solve the proposed transform-based BCS problem formulations (P5.1)-(P5.3). Our algorithms alternate between solving for the sparse codes $\{b_j\}$ (*sparse coding step*), transform W (*transform update step*), and image x (*image update step*), with the other variables kept fixed. We typically alternate a few times between the sparse coding and transform update steps, before performing one image update step. In the following, we describe the three main steps in detail. We show that each of the steps has a simple solution, that can be computed cheaply in practical applications such as MRI.

5.3.1.1 Sparse Coding Step

The sparse coding step of our algorithm for Problem (P5.1) (or (P5.2)) involves the following optimization problem:

$$\min_{B} \sum_{j=1}^{N} \|WR_j x - b_j\|_2^2 \quad s.t. \quad \sum_{j=1}^{N} \|b_j\|_0 \leq s. \tag{5.4}$$

Now, let $Z \in \mathbb{C}^{n \times N}$ be the matrix with the transformed (vectorized) patches $WR_j x$ as its columns. Then, using this notation, Problem (5.4) can be rewritten as follows, where $\|B\|_0$ denotes the number of non-zeros in the sparse code matrix B, and $\|\cdot\|_F$ denotes the standard Frobenius norm.

$$\min_{B} \|Z - B\|_F^2 \quad s.t. \quad \|B\|_0 \leq s. \tag{5.5}$$

The above problem is to project Z onto the non-convex set $\left\{B \in \mathbb{C}^{n \times N} : \|B\|_0 \leq s\right\}$ of matrices that have sparsity $\leq s$, which we call the s-ℓ_0 ball. The optimal projection $\hat{B}$ is easily computed by zeroing out all but the s coefficients of largest magnitude in Z. We denote this operation by $\hat{B} = H_s(Z)$, where $H_s(\cdot)$ is the corresponding projection operator. In case, there is more than one choice for the s elements of largest magnitude in Z, then $H_s(Z)$ is chosen as the projection for which the indices of these s elements are the lowest possible.

In the case of Problem (P5.3), the sparse coding step involves the following unconstrained (and non-convex) optimization problem:

$$\min_{B} \sum_{j=1}^{N} \left\{\|WR_j x - b_j\|_2^2 + \eta^2 \|b_j\|_0\right\} \tag{5.6}$$

which can be rewritten as

$$\min_{B} \|Z - B\|_F^2 + \eta^2 \|B\|_0 \tag{5.7}$$

The optimal solution $\hat{B}$ in this case is obtained as $\hat{B} = \hat{H}_\eta^1(Z)$, with the hard-thresholding operator $\hat{H}_\eta^1(\cdot)$ defined as follows, where the subscript ij

indexes matrix entries (i for row and j for column).

$$\left(\hat{H}_{\eta}^{1}(Z)\right)_{ij} = \begin{cases} 0 & , \ |Z_{ij}| < \eta \\ Z_{ij} & , \ |Z_{ij}| \geq \eta \end{cases} \tag{5.8}$$

The optimal solution to Problem (5.7) is not unique when the condition $|Z_{ij}| = \eta$ is satisfied for some i, j (cf. Page 3 of [96] for a similar scenario and an explanation). The definition in (5.8) chooses one of the multiple optimal solutions in this case.

5.3.1.2 Transform Update Step

Here, we solve for W in the proposed formulations, with the other variables kept fixed. In the case of Problems (P5.1) and (P5.3), this involves the following optimization problem

$$\min_{W} \sum_{j=1}^{N} \|WR_j x - b_j\|_2^2 + 0.5\lambda \|W\|_F^2 - \lambda \log |\det W| \tag{5.9}$$

Now, let $X \in \mathbb{C}^{n \times N}$ be the matrix with the vectorized patches $R_j x$ as its columns, and recall that B is the matrix of codes b_j. Then, Problem (5.9) becomes

$$\min_{W} \|WX - B\|_F^2 + 0.5\lambda \|W\|_F^2 - \lambda \log |\det W| \tag{5.10}$$

An analytical solution for this problem has been recently derived [6, 96], and is stated in the following proposition. It is expressed in terms of an appropriate singular value decomposition (SVD). We let $M^{\frac{1}{2}}$ denote the positive definite square root of a positive definite matrix M.

Proposition 5. *Given $X \in \mathbb{C}^{n \times N}$, $B \in \mathbb{C}^{n \times N}$, and $\lambda > 0$, factorize $XX^H + 0.5\lambda I$ as LL^H, with $L \in \mathbb{C}^{n \times n}$. Further, let $L^{-1}XB^H$ have a full SVD of $V\Sigma R^H$. Then, a global minimizer for the transform update step* (5.10) *is*

$$\hat{W} = 0.5R\left(\Sigma + \left(\Sigma^2 + 2\lambda I\right)^{\frac{1}{2}}\right)V^H L^{-1} \tag{5.11}$$

The solution is unique if and only if XB^H is non-singular. Furthermore, the solution is invariant to the choice of factor L.

Proof. : See the proof of Proposition 1 of [96], particularly the discussion following that proof. □

The factor L in Proposition 5 can for example be the factor L in the Cholesky factorization $XX^H + 0.5\lambda I = LL^H$, or the eigenvalue decomposition (EVD) square root of $XX^H + 0.5\lambda I$. The closed-form solution (5.11) is nevertheless invariant to the particular choice of L. Although in practice the SVD, and the square root of non-negative scalars are computed using iterative methods, we will assume in the convergence analysis in this chapter, that the solution (5.11) (as well as later ones that involve such computations) is computed exactly. In practice, standard numerical methods are guaranteed to quickly provide machine precision accuracy for the SVD or other (aforementioned) computations.

In the case of Problem (P5.2), the transform update step involves the following problem

$$\min_{W} \|WX - B\|_F^2 \;\; s.t. \;\; W^H W = I. \tag{5.12}$$

The solution to the above problem can be expressed as follows (see [6], or Proposition 2 of [96]).

Proposition 6. *Given $X \in \mathbb{C}^{n\times N}$ and $B \in \mathbb{C}^{n\times N}$, let XB^H have a full SVD of $U\Sigma V^H$. Then, a global minimizer in* (5.12) *is*

$$\hat{W} = VU^H \tag{5.13}$$

The solution is unique if and only if XB^H is non-singular.

5.3.1.3 Image Update Step: General Case

In this step, we solve Problems (P5.1)-(P5.3) for the image x, with the other variables fixed. This involves the following optimization problem:

$$\min_{x} \sum_{j=1}^{N} \|WR_j x - b_j\|_2^2 + \nu \|Ax - y\|_2^2 \;\; s.t. \;\; \|x\|_2 \leq C. \tag{5.14}$$

Problem (5.14) is a least squares problem with an ℓ_2 (alternatively, squared ℓ_2) constraint. It can be solved exactly by using the Lagrange multiplier method. The standard Karush-Kuhn-Tucker (KKT) conditions are both necessary and sufficient conditions for optimality in this case. The corresponding Lagrangian formulation is

$$\min_x \sum_{j=1}^{N} \|WR_jx - b_j\|_2^2 + \nu \|Ax - y\|_2^2 + \mu \left(\|x\|_2^2 - C\right) \tag{5.15}$$

where $\mu \geq 0$ is the Lagrange multiplier. The solution to (5.15) satisfies the following Normal Equation

$$\left(\sum_{j=1}^{N} R_j^T W^H W R_j \ + \ \nu \, A^H A + \mu I\right) x = \sum_{j=1}^{N} R_j^T W^H b_j \ + \ \nu \, A^H y \tag{5.16}$$

where $(\cdot)^T$ (matrix transpose) is used instead of $(\cdot)^H$ above for real matrices. The solution to (5.16) is trivially unique for any $\mu > 0$. It is also unique for $\mu = 0$ because the matrix $\sum_j R_j^T W^H W R_j \in \mathbb{C}^{p\times p}$ is positive-definite. In order to see why, consider any $z \in \mathbb{C}^p$. Then, we have

$$z^H \left(\sum_{j=1}^{N} R_j^T W^H W R_j\right) z = \sum_{j=1}^{N} \|WR_jz\|_2^2 \tag{5.17}$$

The right hand side above is non-negative, and moreover zero if and only if $WR_jz = 0 \ \forall \ j$. Since the W in our algorithm is ensured to be invertible, we have that $WR_jz = 0 \ \forall \ j$ if and only if $R_jz = 0 \ \forall \ j$, which implies (assuming that the set of patches in our formulations cover all pixels in the image) that $z = 0$. This ensures the positive-definiteness of $\sum_j R_j^T W^H W R_j$. Thus, the matrix pre-multiplying x in (5.16) is invertible. The unique solution to (5.16) can be found by matrix inversion (for small-sized problems), or by linear conjugate gradients (CG – which is guaranteed to converge to the optimal solution quickly).

In order to solve the original Problem (5.14), the Lagrange multiplier μ in (5.16) must also be chosen optimally. This is done by first computing the EVD of the matrix $\sum_{j=1}^{N} R_j^T W^H W R_j \ + \nu A^H A$ as $U\Sigma U^H$. Since $\sum_{j=1}^{N} R_j^T W^H W R_j$ is positive-definite (by our previous arguments), we have that $\Sigma \succ 0$. Then, denoting as z, the vector $U^H \left(\sum_{j=1}^{N} R_j^T W^H b_j \ + \ \nu \, A^H y\right)$, we have that (5.16) implies $U^H x = (\Sigma + \mu I)^{-1} z$. Therefore,

$$\|x\|_2^2 = \left\|U^H x\right\|_2^2 = \sum_{i=1}^{p} \frac{|z_i|^2}{(\Sigma_{ii} + \mu)^2} \triangleq \tilde{f}(\mu) \tag{5.18}$$

where z_i above denotes the ith entry of vector z. If $\tilde{f}(0) \leq C^2$, then $\hat{\mu} = 0$ is the optimal multiplier. Otherwise, the optimal $\hat{\mu} > 0$. In the latter case, since the function $\tilde{f}(\mu)$ in (5.18) is monotone decreasing for $\mu > 0$, and $\tilde{f}(0) > C^2$, there is a unique multiplier $\hat{\mu} > 0$ such that $\tilde{f}(\hat{\mu}) - C^2 = 0$. The optimal $\hat{\mu}$ is found by using the classical Newton's method (or, alternatively [136]), which in our case is guaranteed to converge to the optimal solution at a quadratic rate. Once the optimal $\hat{\mu}$ is found (to within machine precision), the *unique* solution to the image update Problem (5.14) is the same as the solution to (5.16) with $\mu = \hat{\mu}$.

In practice, when a large value (or, loose estimate) of C is used (for example, in our experiments later in Section 5.5), the optimal solution to (5.14) is typically obtained with the setting $\hat{\mu} = 0$ for the multiplier in (5.15). In this case, the unique minimizer of the objective in (5.14) (obtained with CG) directly satisfies the constraint. Therefore, the additional computations (e.g., EVD) to find the optimal $\hat{\mu}$ can be avoided in this case. An alternative way to find the solution to (5.14) when the optimal $\hat{\mu} \neq 0$, without the EVD computation, is to solve (5.16) repeatedly (by CG) for various μ (tuned in steps) until the $\|x\|_2 = C$ condition is satisfied.

5.3.1.4 Image Update Step: Case of MRI

In certain scenarios, the optimal $\hat{x}$ in (5.14) can be found very efficiently. Here, we consider the case of MRI, where $A = F_u$, the undersampled Fourier encoding matrix. In order to obtain an efficient solution for the $\hat{x}$ in (5.14), we assume that the k-space measurements in MRI are obtained by subsampling on a uniform Cartesian grid. In the following, we first discuss a simple diagonalization approach for the matrix pre-multiplying x in (5.16). We then show that the optimal multiplier $\hat{\mu}$ and the corresponding optimal $\hat{x}$ can be computed without any EVD computations, or CG.

We first diagonalize the matrix $\sum_{j=1}^{N} R_j^T W^H W R_j$ in (5.16). To do this, we make some assumptions on how the overlapping 2D patches are selected from the image(s) in our formulations. First, we assume that periodically positioned, overlapping 2D image patches are used. Furthermore, the patches that begin near the image boundaries are assumed to 'wrap around' on the opposite side of the image [5]. Now, defining the patch *overlap stride* r to be the distance in pixels between corresponding pixel locations in adjacent image patches, it is clear that the setting $r = 1$ results in a maximal set of patches (corresponds to maximal overlap of the patches). When $r = 1$ (and assuming patch 'wrap around'), the following proposition establishes that the matrix $\sum_{j=1}^{N} R_j^T W^H W R_j$ is a 2D circulant matrix, i.e., a *Block Circulant matrix with Circulant Blocks* (abbreviated as BCCB matrix). We let $F \in \mathbb{C}^{p\times p}$ denote the full (2D) Fourier encoding matrix in MRI assumed normalized such that $F^H F = I$.

Proposition 7. *Let $r = 1$, and assume that all 'wrap around' image patches are included. Then, the matrix $\sum_{j=1}^{N} R_j^T W^H W R_j$ in (5.16) is a BCCB matrix with eigenvalue decomposition $F^H \Lambda F$, with $\Lambda \succ 0$.*

Proof. : First, note that $W^H W = \sum_{i=1}^{n} e_i e_i^T W^H W$, where $\{e_i\}_{i=1}^{n}$ are the columns of the $n \times n$ identity matrix. Now, let the ith row of $W^H W$ be denoted as h_i. Then, the matrix $e_i e_i^T W^H W$ is all zero except for its ith row, which is equal to h_i. We denote by G_i, the matrix $\sum_{j=1}^{N} R_j^T \left(e_i e_i^T W^H W\right) R_j$. Then, $\sum_{j=1}^{N} R_j^T W^H W R_j = \sum_{i=1}^{n} G_i$.

We now show that for each i, the matrix G_i when applied to an image $z \in \mathbb{C}^p$ performs the equivalent of 2D circular convolution, that is the vector $G_i z = \sum_{j=1}^{N} R_j^T \left(e_i e_i^T W^H W\right) R_j z$ is equal to the result produced by circularly convolving z with an appropriate 2D filter. In order to see why, we first describe the operation $G_i z$, before proceeding to show its equivalence to 2D circular convolution of z with an appropriately constructed 2D filter.

The product $G_i z = \sum_{j=1}^{N} R_j^T \left(e_i e_i^T W^H W\right) R_j z$ is computed as follows. First, for each j, the vector $\left(e_i e_i^T W^H W\right) R_j z$ is computed. This vector is all zero except (possibly) for its ith entry. The ith entry is obtained as the inner product between h_i and the (vectorized) patch $R_j z$. Next, upon applying R_j^T to $\left(e_i e_i^T W^H W\right) R_j z$, the sparse vector $\left(e_i e_i^T W^H W\right) R_j z$ (that has sparsity ≤ 1) is put back (by R_j^T) at its correct location in the resulting image. Thus, for every j, $R_j^T \left(e_i e_i^T W^H W\right) R_j z$ is an image that is all zero, except for one pixel (a different pixel for each j), where it equals the inner product between

h_i and $R_j z$. Finally, because $r = 1$ (and 'wrap around' patches included), the operation $G_i z = \sum_{j=1}^{N} R_j^T \left(e_i e_i^T W^H W\right) R_j z$ simply populates every pixel in the output with a corresponding inner product (between h_i and a specific patch $R_j z$).

Now, we briefly describe the process of 2D circular convolution, before showing the equivalence of the product $G_i z$ to an appropriate circular convolution. In classical 2D circular convolution, a spatial 2D filter (assumed to be the size of the image) is flipped and translated (1 pixel at a time in 2D) across an image (with 'wrap around' at image edges). For each such translation, an inner product is computed between the translated filter and the image, and this represents a particular circular convolution coefficient. The output of 2D circular convolution is an image whose every pixel is populated with a corresponding inner product (corresponding to a particular translation of the 2D filter).

We now construct a 2D filter, whose circular convolution with $z \in \mathbb{C}^p$ produces the same output as $G_i z$ (described above). First, let us reshape $h_i \in \mathbb{C}^n$ (the i^th row of $W^H W$) into its corresponding $\sqrt{n} \times \sqrt{n}$ patch version denoted as $\hat{h}_i$. Now, let $H_i \in \mathbb{C}^p$ be an image that is all zero, except for a $\sqrt{n} \times \sqrt{n}$ central portion which equals $\hat{h}_i$. The $\hat{h}_i$ is positioned within H_i such that the i^{th} entry of h_i (which is the vectorized $\hat{h}_i$) is the center pixel in H_i. Now, the required 2D filter, which when circularly convolved with z produces $G_i z$, is simply the spatially flipped H_i, which we denote as $\tilde{H}_i$.

In our final argument, we now show that the result obtained by circular convolution between $\tilde{H}_i$ and z is the same as that obtained by the product $G_i z$ (described above). Specifically, because the support of the filter $\tilde{H}_i$ is restricted to a $\sqrt{n} \times \sqrt{n}$ patch window, then, upon considering a flipped (the flipped version is just H_i) and (a specific) translated version of $\tilde{H}_i$ in 2D (with 'wrap around') and taking its inner product with the image z, we obtain the same result as obtained by an inner product between h_i and a specific (vectorized) patch $R_j z$. The circular convolution between $\tilde{H}_i$ and z places such inner products at the same pixels (because of the way H_i, or $\tilde{H}_i$, is defined) in the output as $G_i z$ does. Therefore, the circular convolution between $\tilde{H}_i$ and z is equivalent to the product $G_i z$.

Now, it follows from standard results regarding 2D circular convolution that each G_i is a BCCB matrix. Since $\sum_{j=1}^{N} R_j^T W^H W R_j = \sum_{i=1}^{n} G_i$ is a sum of BCCB matrices, it is therefore a BCCB matrix as well. Now, from standard results regarding BCCB matrices, we know that $\sum_{j=1}^{N} R_j^T W^H W R_j$ has an EVD of $F^H \Lambda F$, where Λ is a diagonal matrix of eigenvalues. It was established in Section 5.3.1.3 that $\sum_{j=1}^{N} R_j^T W^H W R_j$ is positive-definite. Therefore, $\Lambda \succ 0$ holds. $\square$

Proposition 7 guarantees that the matrix $\sum_{j=1}^{N} R_j^T W^H W R_j$ in (5.16) is diagonalizable by the Fourier basis. We now show how the Λ in Proposition 7 can be computed efficiently. First, in Proposition 7, if W were a unitary matrix, then the matrix $\sum_{j=1}^{N} R_j^T W^H W R_j = \sum_{j=1}^{N} R_j^T R_j$ is simply the identity scaled by a factor n (cf. [5]). In this case, $\Lambda = nI$. Second, when W is non-unitary, let us assume that the Fourier matrix F is arranged so that its first column is the constant DC column (with entries $= 1/\sqrt{p}$). Then, the diagonal of Λ in this case is obtained efficiently (via FFTs) as $\sqrt{p} F a_1$, where a_1 is the first column of the matrix $\sum_{j=1}^{N} R_j^T W^H W R_j$. Now, the first column of $\sum_{j=1}^{N} R_j^T W^H W R_j$ can itself be easily computed by applying this operator (using simple patch-based operations) on an image $z \in \mathbb{C}^p$ that has a one in its first entry (first row, first column) and zeros elsewhere. Note that since the image z is extremely sparse, a_1 is computed at a very low cost.

Empowered with the diagonalization result of Proposition 7, we simplify equation (5.16) for MRI, by rewriting it as

$$\begin{aligned} \Big(F \textstyle\sum_{j=1}^{N} R_j^T W^H W R_j F^H \; + \; \nu \, F F_u^H F_u F^H + \mu I \Big) \, Fx \\ = F \textstyle\sum_{j=1}^{N} R_j^T W^H b_j \; + \; \nu \, F F_u^H y \end{aligned} \tag{5.19}$$

where we have applied the operator F to both sides of (5.16), and used the decomposition $I = F^H F$. All p-dimensional vectors (vectorized images) in (5.19) are in Fourier or k-space. Vector $F F_u^H y \in \mathbb{C}^p$ represents the zero-filled (or, zero padded) k-space measurements. The matrix $F F_u^H F_u F^H$ is a diagonal matrix consisting of ones and zeros, with the ones at those diagonal entries that correspond to sampled locations in k-space. Based on Proposition 7 ($r = 1$), the matrix $F \sum_{j=1}^{N} R_j^T W^H W R_j F^H = \Lambda$ is diagonal. Therefore, the matrix pre-multiplying Fx in (5.19) is diagonal and invertible. Denoting the diagonal of Λ by $\delta \in \mathbb{R}^p$ (all positive vector), and $S \triangleq F \sum_{j=1}^{N} R_j^T W^H b_j$, we have that the solution to (5.19) for fixed μ is

$$Fx_\mu \left(k_x, k_y\right) = \begin{cases} \frac{S(k_x, k_y)}{\delta(k_x, k_y) + \mu} & , \, (k_x, k_y) \notin \Omega \\ \frac{S(k_x, k_y) + \nu \, S_0(k_x, k_y)}{\delta(k_x, k_y) + \nu + \mu} & , \, (k_x, k_y) \in \Omega \end{cases} \tag{5.20}$$

where (k_x, k_y) indexes k-space locations, $S_0 = FF_u^H y$, and Ω represents the subset of k-space that has been sampled. Equation (5.20) provides a closed-form solution to the Lagrangian Problem (5.15) for CS MRI, with $Fx_\mu(k_x, k_y)$ representing the optimal updated value (for a particular μ) in k-space at location (k_x, k_y).

The function $\tilde{f}(\mu)$ in (5.18) now has a simple form (no EVD needed) as

$$\tilde{f}(\mu) = \left\| Fx \right\|_2^2 = \sum_{(k_x,k_y)\notin\Omega} \frac{|S(k_x,k_y)|^2}{(\delta(k_x,k_y)+\mu)^2} + \sum_{(k_x,k_y)\in\Omega} \frac{|S(k_x,k_y)+\nu\, S_0(k_x,k_y)|^2}{(\delta(k_x,k_y)+\nu+\mu)^2} \tag{5.21}$$

We check if $\tilde{f}(0) \leq C^2$ first, before applying Newton's method to solve $\tilde{f}(\hat{\mu}) = C^2$. The optimal $\hat{x}$ in (5.14) is obtained via a 2D Inverse FFT of the updated $Fx_{\hat{\mu}}$ in (5.20).

The overall Algorithms A1, A2, and A3 corresponding to the BCS Problems (P5.1), (P5.2), and (P5.3) respectively, are shown in Fig. 5.1.

5.3.2 Computational Properties

Algorithms A1, A2, and A3 involve the steps of sparse coding, transform update, and image update. We now briefly discuss the computational costs of each of these steps.

First, in each outer iteration of our Algorithms A1 and A3, we require $O(n^2N)$ operations to compute the matrix $XX^H + 0.5\lambda I$, where X has the image patches as its columns. The computation of the inverse square root L^{-1} requires only $O(n^3)$ operations, where $n \ll N$ typically.

The cost of the sparse coding step in our algorithms is dominated by the computation of the matrix $Z = WX$ in (5.5) (for Algorithms A1, A2) or (5.7) (for Algorithm A3), and therefore scales as $O(n^2N)$. Notably, the projection onto the s-ℓ_0 ball in (5.5) costs only $O(nN \log N)$ operations, when employing sorting [9], with $\log N \ll n$ typically. Alternatively, in the case of (5.7), the hard thresholding operation costs only $O(nN)$ comparisons.

The cost of the transform update step of our algorithms is dominated by the computation of the matrix XB^H. Since B is sparse, XB^H is computed with $\alpha n^2 N$ multiply-add operations, where $\alpha < 1$ is the fraction of non-zeros in B. The rest of the computations in the transform update step (see Fig. 5.1) take up only $O(n^3)$ operations, where $n \ll N$ typically.

We now discuss the cost of the image update step discussed in Section 5.3.1.4, for the specific case of MRI. We assume $r = 1$, and that the patches 'wrap around', which implies that $N = p$ (i.e., number of patches equals number of image pixels). The computational cost here is dominated by the computation of the term $\sum_{j=1}^{N} R_j^T W^H b_j$ in the normal equation (5.16), which takes $\beta n^2 N$ multiply-add operations, with β being a (small) fraction typically. All other operations in the image update step have a much lower computational cost. The various FFT and IFFT operations cost only $O(N \log N)$ operations, where $\log N \ll n$ typically. The Newton's method to compute the optimal multiplier $\hat{\mu}$ is only used when $\mu = 0$ is non-optimal. In the latter case, Newton's method takes up $O(N\tilde{J})$ operations, with $\tilde{J}$ being the number of iterations (typically small, and independent of n) of Newton's method.

Based on the preceding arguments, it is easy to observe that the total cost per (outer) iteration of the algorithms in Fig. 5.1 on page 140 scales (for MRI) as $O(n^2 NL)$. Now, the recent synthesis dictionary-based BCS method called DLMRI [5] learns a dictionary $D \in \mathbb{C}^{n \times K}$ from CS MRI measurements by solving Problem (P5.0). For this scheme, the computational cost per outer iteration scales as $O(NKns\hat{J})$ [9], where $\hat{J}$ is the number of (inner) iterations of dictionary learning (using the K-SVD algorithm [17]), and the other notations are the same as in (P5.0). Assuming that $K \propto n$, and that the synthesis sparsity $s \propto n$, we have that the cost per iteration of DLMRI scales as $O(n^3 N\hat{J})$. Thus, the per-iteration computational cost of the proposed BCS schemes is much lower (lower in order by factor n assuming $L \sim \hat{J}$) than that for synthesis dictionary-based BCS. This gap in computations is amplified for higher-dimensional imaging applications such as 3D or 4D imaging, where the size of the 3D or 4D patches is typically much bigger than in the case of 2D imaging.

As illustrated in our experiments in Section 5.5, the proposed BCS algorithms converge in few iterations in practice. Therefore, the per-iteration computational advantages over synthesis dictionary-based BCS also typically translate to a net computational advantage in practice.

5.4 Convergence Results

Here, we present convergence guarantees for Algorithms A1, A2, and A3, that solve Problems (P5.1), (P5.2), and (P5.3), respectively. These problems are highly non-convex. Notably they involve either a ℓ_0 penalty or constraint for sparsity, a non-convex transform regularizer or constraint, and the term $\|WR_jx - b_j\|_2^2$ that is a highly non-convex function involving the product of unknown matrices and vectors. The proposed algorithms for Problems (P5.1)-(P5.3) are block coordinate descent-type algorithms. We previously discussed in Proposition 4, the issue of (noiseless) identifiability of the underlying image by solving the proposed problems. We are now interested in understanding whether the proposed algorithms converge to a minimizer of the corresponding problems, or whether they possibly get stuck in non-stationary points. Due to the high degree of non-convexity involved here, standard results on convergence of block coordinate descent methods (e.g., [137]) do not apply here.

Very recent works on the convergence of block coordinate descent-type algorithms (e.g., [138], or the Block Coordinate Variable Metric Forward-Backward algorithm [139]) prove convergence of the iterate sequence (for specific algorithm) to a critical point of the objective. However, these works make numerous assumptions, some of which can be easily shown to be violated for the proposed formulations (for example, the term $\sum_{j=1}^{N} \|WR_jx - b_j\|_2^2$ in the objectives of our formulations, although differentiable, violates the L-Lipschitzian gradient property described in Assumption 2.1 of [139]).

In fact, in certain simple scenarios (e.g., when $y = 0$), one can easily derive non-convergent iterate sequences for the Algorithms in Fig. 5.1. Non-convergence mainly arises for the transform or sparse code sequences (rather than the image sequence) due to the fact that the optimal solutions in the sparse coding or transform update steps may be non-unique.

In this work, we provide some convergence guarantees for the proposed BCS approaches, where the only assumption is that the various steps in our algorithms are solved exactly. (Recall that machine precision is guaranteed in practice.) We first introduce some relevant background and notations before stating our main results.

5.4.1 Preliminaries

We first list some definitions that will be used in our analysis.

Definition 1. *For a function $\phi : \mathbb{R}^q \mapsto (-\infty, +\infty]$, its domain is defined as* $\mathrm{dom}\phi = \{z \in \mathbb{R}^q : \phi(z) < +\infty\}$. *Function ϕ is proper if* $\mathrm{dom}\phi$ *is nonempty.*

Next, we define the notion of Fréchet sub-differential for a function as follows [140, 141]. The norm and inner product notations used below correspond to the Euclidean ℓ_2 settings.

Definition 2. *Let $\phi : \mathbb{R}^q \mapsto (-\infty, +\infty]$ be a proper function and let $z \in$* $\mathrm{dom}\phi$. *The Fréchet sub-differential of the function ϕ at z is the following set:*

$$\hat{\partial}\phi(z) \triangleq \left\{h \in \mathbb{R}^q : \liminf_{b \to z, b \neq z} \tfrac{1}{\|b-z\|} \left(\phi(b) - \phi(z) - \langle b - z, h\rangle\right) \geq 0\right\} \tag{5.22}$$

If $z \notin$ $\mathrm{dom}\phi$, *then $\hat{\partial}\phi(z) = \emptyset$. The sub-differential of ϕ at z is defined as*

$$\partial\phi(z) \triangleq \left\{\tilde{h} \in \mathbb{R}^q : \exists z_k \to z, \phi(z_k) \to \phi(z), h_k \in \hat{\partial}\phi(z_k) \to \tilde{h}\right\}. \tag{5.23}$$

The above definition implies that $\hat{\partial}\phi(z) \subset \partial\phi(z)$ for each $z \in \mathbb{R}^q$, where the first set is convex and closed while the second one is closed [140]. A necessary condition for $z \in \mathbb{R}^q$ to be a minimizer of the function $\phi : \mathbb{R}^q \mapsto (-\infty, +\infty]$ is that z is a *critical point* of ϕ, i.e., $0 \in \partial\phi(z)$. If ϕ is a convex function, this condition is also sufficient. Critical points can be thought of as "generalized stationary points" [140].

We say that a sequence $\{a^t\}$ with $a^t \in \mathbb{C}^q$ has an accumulation point a, if there is a subsequence that converges to a.

5.4.2 Notations

Problems (P5.1) and (P5.2) have the constraint $\sum_{j=1}^{N} \|b_j\|_0 \leq s$, which can instead be added as a penalty in the respective objectives by using a barrier function $\psi(B)$ (which takes the value $+\infty$ when the sparsity constraint is violated, and is zero otherwise). Problem (P5.2) also has the constraint $W^H W = I$, which can be equivalently added as a penalty in the objective of (P5.2) by using the barrier function $\varphi(W)$, which takes the value $+\infty$

when the unitary constraint is violated, and is zero otherwise. Finally, the constraint $\|x\|_2 \leq C$ in our formulations is replaced by the barrier function penalty $\chi(x)$. With these modifications, all the proposed problem formulations can be written in an unconstrained form. The objectives of (P5.1), (P5.2), and (P5.3), are then respectively denoted as

$$g(W, B, x) = \sum_{j=1}^{N} \|WR_j x - b_j\|_2^2 + \nu \|Ax - y\|_2^2 + \lambda\, Q(W) + \psi(B) + \chi(x) \tag{5.24}$$

$$u(W, B, x) = \sum_{j=1}^{N} \|WR_j x - b_j\|_2^2 + \nu \|Ax - y\|_2^2 + \psi(B) + \varphi(W) + \chi(x) \tag{5.25}$$

$$v(W, B, x) = \sum_{j=1}^{N} \|WR_j x - b_j\|_2^2 + \nu \|Ax - y\|_2^2 + \lambda\, Q(W) + \eta^2 \sum_{j=1}^{N} \|b_j\|_0 + \chi(x) \tag{5.26}$$

It is easy to see that (cf. [96] for a similar statement and justification) the unconstrained minimization problem involving the objective $g(W, B, x)$ (alternatively, $u(W, B, x)$, or $v(W, B, x)$) is exactly equivalent to the corresponding constrained formulation (P5.1) (alternatively, (P5.2), or (P5.3)), in the sense that the minimum objective values as well as the set of minimizers of the two formulations are identical. The proposed algorithms are block coordinate descent algorithms for both the corresponding constrained and unconstrained formulations above.

Since the functions g, u, and v accept complex-valued (input) arguments, we will compute all derivatives or sub-differentials (Definition 2) of these functions with respect to the (real-valued) real and imaginary parts of the variables (W, B, x). Note that the functions g, u, and v are proper (we set the negative log-determinant penalty to be $+\infty$ wherever $\det W = 0$) and lower semi-continuous. For the algorithms in Fig. 5.1, we denote the iterates (outputs) in each outer iteration t by the set (W^t, B^t, x^t).

For a matrix H, we let $\rho_j(H)$ denote the magnitude of the j^{th} largest element (magnitude-wise) of the matrix H. For some matrix E, $\|E\|_\infty \triangleq \max_{i,j} |E_{ij}|$. Finally, $Re(A)$ denotes the real part of some scalar or matrix A.

5.4.3 Main Results

The following theorem provides the convergence result for Algorithm A1 that solves Problem (P5.1). We assume that the initial estimates (W^0, B^0, x^0) satisfy all problem constraints.

Theorem 3. *Let $\{W^t, B^t, x^t\}$ denote the iterate sequence generated by Algorithm A1 with measurements $y \in \mathbb{C}^m$ and initial (W^0, B^0, x^0). Then, the objective sequence $\{g^t\}$ with $g^t \triangleq g\left(W^t, B^t, x^t\right)$ is monotone decreasing, and converges to a finite value, say $g^* = g^*(W^0, B^0, x^0)$. Moreover, the iterate sequence $\{W^t, B^t, x^t\}$ is bounded, and all its accumulation points are equivalent in the sense that they achieve the exact same value g^* of the objective. The sequence $\{a^t\}$ with $a^t \triangleq \left\|x^t - x^{t-1}\right\|_2$, converges to zero. Finally, every accumulation point (W, B, x) of $\{W^t, B^t, x^t\}$ is a critical point of the objective g satisfying the following partial global optimality conditions:*

$$x \in \underset{\tilde{x}}{\arg\min}\ g\left(W, B, \tilde{x}\right) \tag{5.27}$$

$$W \in \underset{\tilde{W}}{\arg\min}\ g\left(\tilde{W}, B, x\right) \tag{5.28}$$

$$B \in \underset{\tilde{B}}{\arg\min}\ g\left(W, \tilde{B}, x\right) \tag{5.29}$$

Each accumulation point (W, B, x) also satisfies the following partial local optimality conditions:

$$g(W + dW, B + \Delta B, x) \geq g(W, B, x) = g^* \tag{5.30}$$

$$g(W, B + \Delta B, x + \tilde{\Delta} x) \geq g(W, B, x) = g^* \tag{5.31}$$

The conditions each hold for all $\tilde{\Delta} x \in \mathbb{C}^p$, and all sufficiently small $dW \in \mathbb{C}^{n\times n}$ satisfying $\|dW\|_F \leq \epsilon'$ for some $\epsilon' > 0$ that depends on the specific W, and all $\Delta B \in \mathbb{C}^{n\times N}$ in the union of the following regions R1 and R2, where $X \in \mathbb{C}^{n\times N}$ is the matrix with $R_j x$, $1 \leq j \leq N$, as its columns.

R1. The half-space $Re\left(tr\left\{(WX - B)\Delta B^H\right\}\right) \leq 0$.

R2. The local region defined by $\|\Delta B\|_\infty < \rho_s(WX)$.

Furthermore, if $\|WX\|_0 \leq s$, then ΔB can be arbitrary.

Theorem 3 establishes that for each initial point (W^0, B^0, x^0), the iterate sequence in Algorithm A1 converges to an equivalence class of accumulation points. Specifically, every accumulation point corresponds to the same value $g^* = g^*(W^0, B^0, x^0)$ of the objective. The exact value of g^* could vary with initialization. Importantly, the equivalent accumulation points are all critical points as well as *at least* partial minimizers of the objective $g(W, B, x)$, in the following sense. Every accumulation point (W, B, x) is a partial global minimizer of $g(W, B, x)$ with respect to each of W, B, and x, as well as a partial local minimizer of $g(W, B, x)$ with respect to (W, B), and (B, x), respectively. Therefore, we have the following corollary to Theorem 3.

Corollary 6. *For each* (W^0, B^0, x^0)*, the iterate sequence in Algorithm A1 converges to an equivalence class of critical points, that are also partial minimizers satisfying* (5.27)*,* (5.28)*,* (5.29)*,* (5.30)*, and* (5.31)*.*

Conditions (5.30) and (5.31) in Theorem 3 hold true not only for local (or small) perturbations of the sparse code matrix (accumulation point) B, but also for arbitrarily large perturbations of the sparse codes in a half space, as defined by region R1. Furthermore, the partial optimality condition (5.31) also holds for arbitrary perturbations $\tilde{\Delta}x$ of x.

Theorem 3 also says that $\|x^t - x^{t-1}\|_2 \to 0$. This is a necessary but not sufficient condition for convergence of the entire sequence $\{x^t\}$.

The following corollary to Theorem 3 also holds, where 'globally convergent' refers to convergence from any initialization.

Corollary 7. *Algorithm A1 is globally convergent to a subset of the set of critical points of the non-convex objective* $g(W, B, x)$*. The subset includes all critical points* (W, B, x)*, that are at least partial global minimizers of* $g(W, B, x)$ *with respect to each of* W*,* B*, and* x*, as well as partial local minimizers of* $g(W, B, x)$ *with respect to* (W, B)*, and* (B, x)*, respectively.*

Theorem 3 holds for Algorithm A1 irrespective of the number of inner alternations L, between transform update and sparse coding, within each outer algorithm iteration in Fig. 5.1. In practice, we have observed that a larger value of L (particularly in initial algorithm iterations) enables Algorithm A1 to be insensitive to the initial (even, badly chosen) values of W^0 and B^0.

The convergence results for Algorithms A2 or A3 are quite similar to that for Algorithm A1. The following two Theorems briefly state the results for Algorithms A3 and A2, respectively.

Theorem 4. *Theorem 3 applies to Algorithm A3 and the corresponding objective $v(W, B, x)$ as well, except that the set of perturbations $\Delta B \in \mathbb{C}^{n\times N}$ in Theorem 3 is restricted to $\|\Delta B\|_\infty < \eta/2$ for Algorithm A3.*

Theorem 5. *Theorem 3, except for the condition* (5.30)*, applies to Algorithms A2 and the corresponding objective $u(W, B, x)$ as well.*

Note that owing to Theorems 4 and 5, results similar to Corollaries 6 and 7 also apply for Algorithms A2 and A3, respectively. The proofs of the stated convergence theorems are provided in Appendix C.

5.5 Numerical Experiments

5.5.1 Framework

Here, we study the usefulness of the proposed sparsifying transform-based blind compressed sensing framework for the CS MRI application [2]. The in vivo MR data used in these experiments were kindly provided by Prof. Michael Lustig, UC Berkeley. We simulate various undersampling patterns in k-space including variable density 2D random sampling [3] [122, 5], and Cartesian sampling with (variable density) random phase encodes (1D random). We employ Problem (P5.1) and the corresponding Algorithm A1 to reconstruct images from undersampled measurements in the experiments here. Our reconstruction method is referred to as Transform Learning MRI (TLMRI).

First, we illustrate the practical convergence behavior of TLMRI. We also compare the reconstructions provided by the TLMRI method to those provided by the following schemes: 1) the Sparse MRI method of Lustig et al. [3], that utilizes Wavelets and Total Variation as *fixed* sparsifying transforms;

[2]We have also proposed another sparsifying transform-based BCS MRI method recently [104]. However, the latter approach involves many more parameters (e.g., error thresholds to determine patch-wise sparsity levels), which may be hard to tune in practice. In contrast, the methods proposed in this chapter involve only a few relatively easy to set parameters.

[3]This sampling scheme is feasible when data corresponding to multiple image slices are jointly acquired, and the phase encode direction is chosen perpendicular to the image plane.

2) the DLMRI method [5] that learns adaptive overcomplete sparsifying dictionaries; and 3) the PBDWS method [4]. The PBDWS method is a very recent *partially* adaptive sparsifying-transform based compressed sensing reconstruction method (that uses redundant Wavelets and trained patch-based geometric directions). It has been shown to perform better than the earlier PBDW method [125].

For the Sparse MRI and PBDWS methods, we used the software implementations available from the respective authors' websites [142, 143]. We used the built-in parameter settings in those implementations, which performed well in our experiments. Specifically, for the PBDWS method, the shift invariant discrete Wavelet transform (SIDWT) based reconstructed image is used as the *guide* (initial) image [4, 143]. The implementation of the DLMRI algorithm that solves Problem (P5.0) is also available online [144]. For this scheme, we work with 6×6 patches ($n = 36$) as suggested in [5] [4], and learn a four-fold overcomplete synthesis dictionary ($K = 144$) using 25 iterations of the algorithm. We set $r = 1$, and use 14400 (found empirically [5]) randomly selected patches during the dictionary learning step (executed for 20 iterations) of the algorithm. We employ both a maximum sparsity level (of $s = 7$ per patch) and an error threshold (for sparse coding) during the dictionary learning step. The ℓ_2 error threshold per patch varies linearly from 0.34 to 0.15 over the DLMRI iterations. These parameter settings (all other parameters are set as per the indications in the DLMRI-Lab toolbox [144]) were observed to work well for the DLMRI algorithm.

The parameters for TLMRI (with Algorithms A1) are set to $n = 36$, $r = 1$ (with patch wrap around), $\nu = 3.81$, $L = 1$, $\lambda_0 = 0.2$, $C = 10^5$, and $s = 0.034 \times nN$ (this corresponds to an average sparsity level per patch of $0.034 \times n$, or 3.4% sparsity), where $N = 512^2$. The initial transform estimate W^0 is the (simple) patch-based 2D DCT [9], and the initial image x^0 is set to be the standard zero-filling Fourier reconstruction. The initial sparse code settings are the solution to (5.4), for the given (W^0, x^0). Our TLMRI implementation was coded in Matlab version R2013a. Note that this implementation has not been optimized for efficiency. All simulations in this

[4]The reconstruction quality improves slightly with a larger patch size, but with a substantial increase in runtime.

[5]Using a larger training size (> 14400) during the dictionary learning step of the algorithm provides negligible improvement in final image reconstruction quality, while leading to increased runtimes.

work were executed in Matlab. All computations were performed with an Intel Core i5 CPU at 2.5GHz and 4GB memory, employing a 64-bit Windows 7 operating system.

Similar to prior work [5], we quantify the quality of MR image reconstruction using the peak-signal-to-noise ratio (PSNR), and high frequency error norm (HFEN) metrics. The PSNR (expressed in dB) is computed as the scaled (by the factor 20) base-10 logarithm of the ratio of the peak intensity value of some reference image to the root mean square (rms) reconstruction error relative to the reference (rms error is computed here between the reconstructed and reference image magnitudes). In MRI, the reference image is typically the image reconstructed from fully sampled k-space data. The HFEN metric quantifies the quality of reconstruction of edges or finer features. A rotationally symmetric Laplacian of Gaussian (LoG) filter is used, whose kernel is of size 15×15 pixels, and with a standard deviation of 1.5 pixels [5]. HFEN is computed as the ℓ_2 norm of the difference between the LoG filtered reconstructed and reference magnitude images.

5.5.2 Convergence and Learning Behavior

In this experiment, we consider the 512×512 complex-valued reference image shown (only the magnitude of the complex-valued image is displayed) in Fig. 5.2(a) on page 141, that is reconstructed (by inverse 2D FFT) from fully-sampled Cartesian (raw) k-space data. This is an Axial T2-weighted brain image whose raw k-space was obtained using a Fast Spin Echo sequence, with slice thickness 5 mm, TE = 84.64 ms, TR = 4.8 s, FOV = 20 cm, and bandwidth 20.83 kHz. We perform four-fold undersampling of the 2D DFT space of the (normalized) reference. The sampling mask is shown in Fig. 5.2(b). When the TLMRI algorithm is executed using the undersampled data, the objective function converges monotonically and quickly over the iterations as shown in Fig. 5.2(e). The changes between successive iterates $\|x^t - x^{t-1}\|_2$ (Fig. 5.2(g)) converge towards 0. Such convergence was established by Theorem 3, and is indicative (a necessary but not suffficient condition) of convergence of the entire sequence $\{x^t\}$. As far as the performance metrics are concerned, the PSNR metric (Fig. 5.2(f)) increases over the iterations, and the HFEN metric decreases, indicating improving reconstruction quality over the algorithm iterations. These metrics also converge quickly.

The initial zero-filling reconstruction (Fig. 5.2(c)) shows aliasing artifacts that are typical in the undersampled measurement scenario, and has a PSNR of only 28.94 dB. On the other hand, the final TLMRI reconstruction (Fig. 5.2(d)) is much enhanced (by 4 dB), with a PSNR of 32.66 dB. Since Algorithm A1 is guaranteed to converge to the set of critical points of Problem (P5.1), the result in Fig. 5.2(d) suggests that, in practice, the set of critical points may in fact include images that are close to the true image. Note that our previous identifiability result (Proposition 18) in Section 5.2.2 ensured global optimality of the underlying image only in a noiseless (or error-free) scenario. The learned transform W ($\kappa(W) = 1.01$) for this example is shown in Fig. 5.2(h). This is a complex valued transform. Both the real and imaginary parts of W display texture or frequency like structures, that sparsify the patches of the brain image. Our algorithm is thus able to learn this structure and reconstruct the image using only the undersampled measurements.

Figs. 5.2(i) and 5.2(j) provide yet another visualization of the learned transform by showing the magnitude and phase of the transform. Specifically, the phase displays interesting (spatially varying) features, that point out the richness of the learned model.

5.5.3 Comparison to Other Methods

In the following experiments, we execute the TLMRI algorithm for 25 iterations. We also use a lower sparsity level ($< 0.034 \times nN$) during the initial few iterations, which leads to faster convergence. We consider the complex-valued reference in Fig. 5.2(a), and simulate Cartesian and 2D random undersampling of k-space (2D DFT of (normalized) reference) at two different undersampling factors each, namely 4 fold and 7 fold undersampling. Table 5.1 lists the reconstruction PSNRs corresponding to the zero-filling, Sparse MRI, PBDWS, DLMRI, and TLMRI reconstructions for the various cases.

The TLMRI algorithm is seen to provide the best PSNRs (analogous results hold with respect to the HFEN metric not shown in the table) for the various scenarios in Table 5.1. Significant improvements (up to 5.5 dB) are observed over the Sparse MRI method, that uses fixed sparsifying trans-

Sampling Scheme	Undersampling	Zero-filling	Sparse MRI	PBDWS	DLMRI	TLMRI
2D Random	4x	25.30	30.32	32.64	32.91	**33.04**
	7x	25.33	27.34	31.31	31.46	**31.81**
Cartesian	4x	28.94	30.20	32.02	32.46	**32.64**
	7x	27.87	25.53	30.09	30.72	**31.04**

Table 5.1: PSNRs corresponding to the Zero-filling, Sparse MRI [3], PBDWS [4], DLMRI [5], and TLMRI reconstructions, for various sampling schemes and undersampling factors. The best PSNRs are marked in bold.

forms. Moreover, TLMRI provides up to 1 dB improvement in PSNR over the recent (partially adaptive) PBDWS method. Finally, the TLMRI reconstruction quality is somewhat (up to 0.35 dB) better than DLMRI. This is despite the latter using a 4 fold overcomplete (i.e., larger or richer) dictionary. Fig. 5.3 shows the reconstruction errors (i.e., the magnitude of the difference between the magnitudes of the reconstructed and reference images) for the various schemes. The error maps for TLMRI clearly show the smallest image distortions.

The average run times of the Sparse MRI, PBDWS, DLMRI, and TLMRI algorithms in Table 5.1 are 251 seconds, 794 seconds, 2051 seconds, and 211 seconds, respectively. The PBDWS run time includes the time (about 5 mins) taken for computing the initial SIDWT based reconstruction [4]. The TLMRI algorithm is thus the fastest one in Table 5.1, and provides a speedup of about 10x over the synthesis dictionary-based DLMRI, and a speedup of about 4x over the PBDWS method.

In our last example, we consider one image slice (with rich features) from a multislice data acquisition. The 512×512 complex-valued reference image reconstructed from fully-sampled k-space data is shown (only the magnitude of the complex-valued image is displayed) in Fig. 5.4(a). We employ 2D random sampling with 5 fold undersampling (Fig. 5.4(b)) in the Fourier space of the (normalized) reference. The TLMRI reconstruction (Fig. 5.4(c)) in this case is much more freer of artifacts than the DLMRI one (Fig. 5.4(d)). The PSNRs for the Sparse MRI, PBDWS, DLMRI, and TLMRI methods in this case are 27.51 dB, 30.26 dB, 28.54 dB, and 30.47 dB, respectively.

While our results show some (preliminary) potential for the proposed sparsifying transform-based blind compressed sensing framework (for MRI), a much more detailed investigation will be presented elsewhere. Combining the proposed scheme with the patch-based directional Wavelets ideas [125, 4], or extending our framework to learning overcomplete sparsifying transforms (c.f., [134]) could potentially boost BCS performance further.

5.6 Conclusions

In this chapter, we presented a novel sparsifying transform-based framework for blind compressed sensing. Our formulations exploit the (adaptive) transform domain sparsity of overlapping image patches in 2D, or voxels in 3D. The proposed formulations are however highly nonconvex. Our block coordinate descent-type algorithms for solving the proposed problems involve highly efficient update steps. Importantly, our algorithms are guaranteed to converge to the critical points of the objectives defining the proposed formulations. These critical points are also guaranteed to be at least partial local/global minimizers. Our numerical examples showed the usefulness of the proposed scheme for magnetic resonance image reconstruction from highly undersampled k-space data. Our approach while being highly efficient also provides promising MR image reconstruction quality. The usefulness of the proposed blind compressed sensing methods in other inverse problems and imaging applications merits further study.

```
Transform-Based BCS Algorithms A1, A2, and A3
**Inputs:** $y$ - measurements obtained with sensing matrix $A$, $s$ - sparsity, $\lambda$ - weight, $\nu$ - weight, $C$ - energy bound, $L$ - number of inner iterations, $J$ - number of outer iterations.
**Outputs:** $x$ - reconstructed image, $W$ - adapted sparsifying transform, $B$ - matrix with sparse codes of all overlapping patches as columns.
**Initial Estimates:** $\left(W^0, B^0, x^0\right)$.
**For t = 1: J Repeat**
1) Form the matrix $X$ with $R_j x^{t-1}$ as its columns. Compute $L^{-1} = \left(XX^H + 0.5\lambda I\right)^{-1/2}$ for Algorithms A1 and A3. Set $\tilde{B}^0 = B^{t-1}$.
2) **For l = 1: L Repeat**
    (a) **Transform Update Step:**
        i. Set $V\Sigma R^H$ as the full SVD of $L^{-1}X\left(\tilde{B}^{l-1}\right)^H$ for Algorithms A1 and A3, or the full SVD of $X\left(\tilde{B}^{l-1}\right)^H$ for Algorithm A2.
        ii. $\tilde{W}^l = 0.5R\left(\Sigma + \left(\Sigma^2 + 2\lambda I\right)^{\frac{1}{2}}\right)V^H L^{-1}$ for Algorithms A1 and A3, or $\tilde{W}^l = RV^H$ for Algorithm A2.
    (b) **Sparse Coding Step:** $\tilde{B}^l = H_s\left(\tilde{W}^l X\right)$ for Algorithms A1 and A2, or $\tilde{B}^l = \hat{H}^1_\eta\left(\tilde{W}^l X\right)$ for Algorithm A3.
   **End**
3) Set $W^t = \tilde{W}^L$ and $B^t = \tilde{B}^L$. Set $b^t_j$ as the j$^{\text{th}}$ column of $B^t$ $\forall\ j$.
4) **Image Update Step:**
    (a) For generic CS scheme, solve (5.16) for $x^t$ with $\mu = 0$, by linear CG. If $\left\|x^t\right\|_2 > C$,
        i. Compute $U\Sigma U^H$ as EVD of $\sum_{j=1}^N R_j^T\left(W^t\right)^H W^t R_j\ +\ \nu\, A^H A$.
        ii. Compute $z = U^H\left(\sum_{j=1}^N R_j^T\left(W^t\right)^H b^t_j\ +\ \nu\, A^H y\right)$.
        iii. Use Newton's method to find $\hat{\mu}$ such that $\tilde{f}(\hat{\mu}) = C^2$ in (5.18).
        iv. $x^t = U\left(\Sigma + \hat{\mu}I\right)^{-1} z$.
    (b) For MRI, do the following
        i. Compute the image $c = \sum_{j=1}^N R_j^T\left(W^t\right)^H b^t_j$. $S \leftarrow FFT(c)$.
        ii. Compute $a_1$ as the first column of $\sum_{j=1}^N R_j^T\left(W^t\right)^H W^t R_j$.
        iii. Set $\delta \leftarrow \sqrt{p} \times FFT(a_1)$.
        iv. Compute $\tilde{f}(0)$ as per (5.21).
        v. If $\tilde{f}(0) \le C^2$, set $\hat{\mu} = 0$. Else, use Newton's method to solve $\tilde{f}(\hat{\mu}) = C^2$ for $\hat{\mu}$.
        vi. Update $S$ to be the right hand side of (5.20) with $\mu = \hat{\mu}$. $x^t = IFFT(S)$.
**End**
```

Figure 5.1: Algorithms A1-A3 corresponding to Problems (P5.1)-(P5.3), respectively. The superscripts t and l denote the main iterates, and the iterates in the inner alternations between transform update and sparse coding, respectively. The abbreviations FFT and $IFFT$ denote the fast implementations of the normalized 2D DFT and 2D IDFT. For MRI, $r = 1$ is assumed, and the encoding matrix F is assumed normalized, and arranged so that its first column is the constant DC column.

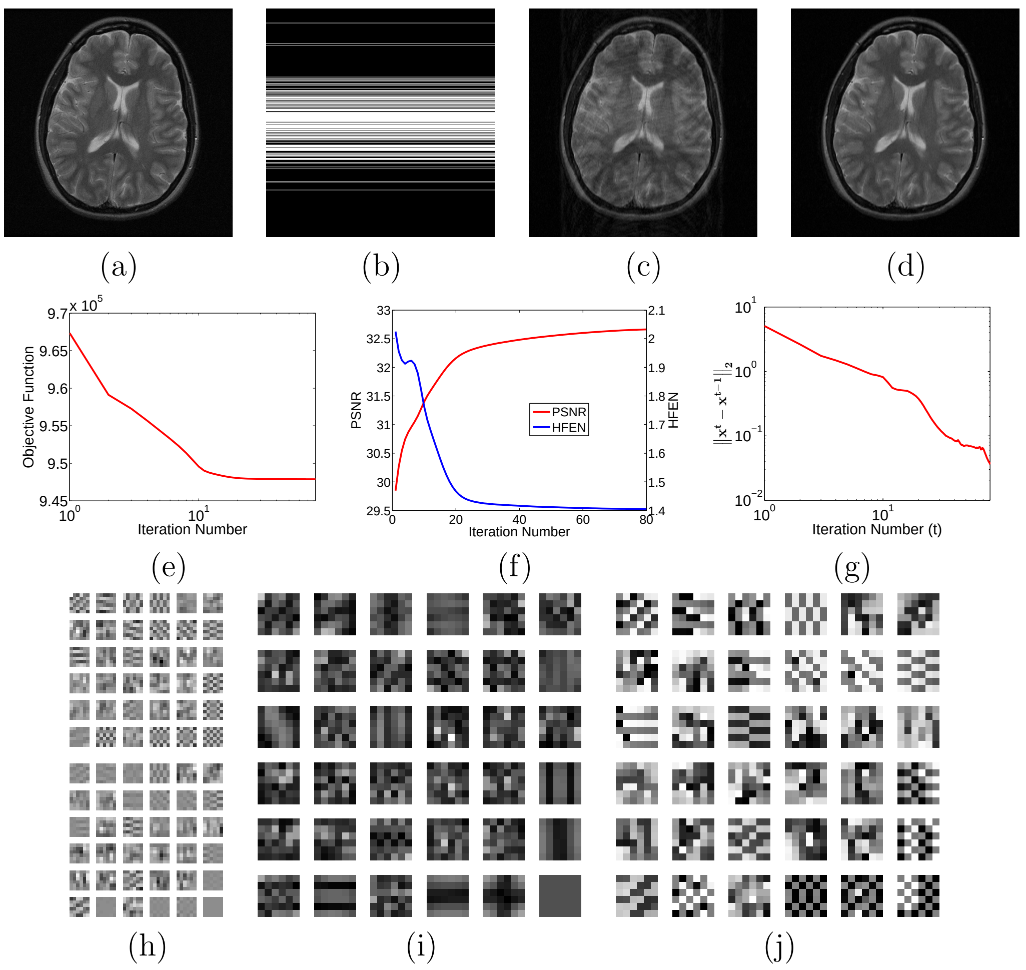
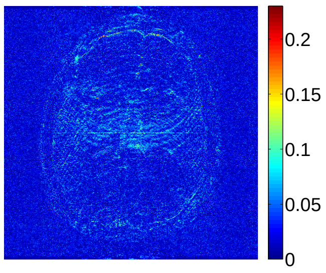
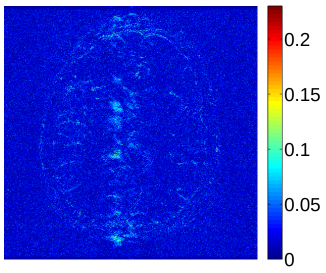


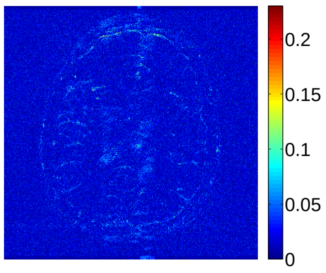
Figure 5.2: Convergence of TLMRI with 4x undersampling: (a) Magnitude of the (reference) image reconstructed from fully-sampled k-space data; (b) sampling mask in k-space; (c) magnitude of the initial zero-filling reconstruction (28.94 dB); (d) magnitude of the TLMRI reconstruction (32.66 dB); (e) objective function; (f) PSNR and HFEN; (g) changes between successive iterates ($\|x^t - x^{t-1}\|_2$); (h) real (top) and imaginary (bottom) parts of the learned W, with the matrix rows shown as patches; and finally the magnitude (i) and phase (j) of the learned W, with the rows shown as patches.



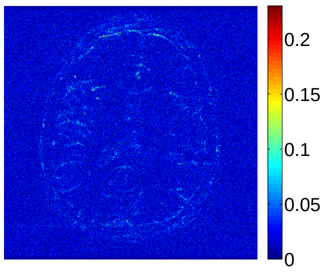
(a)



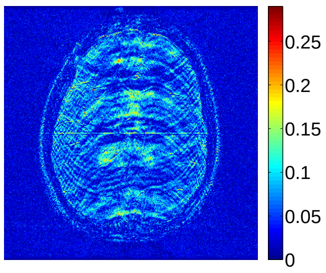
(b)



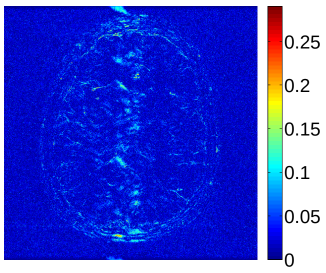
(c)



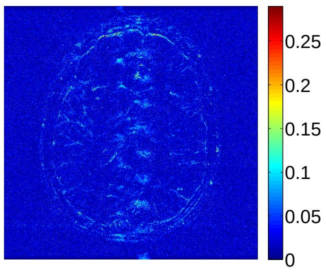
(d)



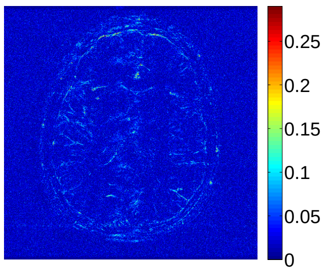
(e)



(f)



(g)



(h)

Figure 5.3: Magnitude of reconstruction error at 4x undersampling (Cartesian) for the following methods: (a) Sparse MRI [3]; (b) PBDWS [4]; (c) DLMRI [5]; and (d) TLMRI. Reconstruction error magnitudes at 7x undersampling (Cartesian) for (e) Sparse MRI [3], (f) PBDWS [4], (g) DLMRI [5], and (h) TLMRI.

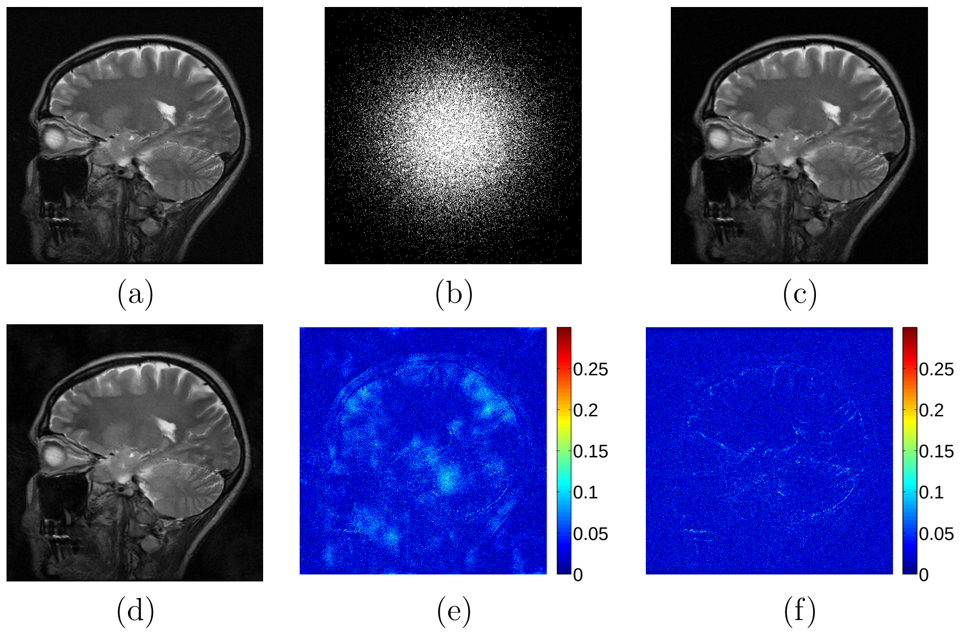


Figure 5.4: 2D random sampling with 5 fold undersampling: (a) Magnitude of the (reference) image reconstructed from fully-sampled k-space data; (b) sampling mask in k-space; (c) magnitude of the TLMRI reconstruction (30.47 dB); (d) magnitude of the DLMRI reconstruction (28.54 dB); (e) magnitude of DLMRI reconstruction error; and (f) magnitude of TLMRI reconstruction error.

CHAPTER 6

ADAPTIVE SAMPLING DESIGN FOR COMPRESSED SENSING MRI

6.1 Introduction

In this chapter, we discuss the design of adaptive sampling schemes for compressed sensing MRI (CSMRI) [1].

Since the k-space measurements in magnetic resonance imaging (MRI) are acquired (or, sampled) sequentially, MRI tends to be a relatively slow imaging modality. Compressed sensing promises accurate image reconstructions for MRI using far fewer measurements than mandated by traditional Nyquist sampling. Various sampling schemes have been proposed for CSMRI such as Cartesian sampling with random phase encodes and pseudo random 2D sampling [3]. Wang et al. [146] approximate pseudo random 2D sampling (for a single 2D image) with bidirectional Cartesian sampling that is realized with a pulse sequence program that switches the directions of phase encoding and frequency encoding during data acquisition. To account for the unequal distribution of signal energy across k-space, Lustig et al. [3] perform variable density random sampling of k-space by drawing sample positions according to a probability density function (pdf). Out of many such candidate patterns, they choose the one that has the lowest mutual coherence with the sparsifying transform. However, the procedure uses an ad-hoc model for the pdf and is nonadaptive. Moreover, finding the optimal sampling scheme that maximizes the incoherence for a given number of samples is a combinatorial problem that is intractable.

Although random sampling has good asymptotic properties and there are theoretical performance guarantees for CS based on mutual coherence, these results are inapplicable for the small sample (matrix) sizes in CSMRI – especially with high subsampling. Existing methods for sampling pattern se-

[1]The material of this chapter was previously presented in [145].

lection in CSMRI may therefore have room for improvement.

In this chapter, we focus on the design of adaptive sampling patterns using training image scans. Such sampling patterns capture the underlying structure in k-space to provide superior reconstructions for CSMRI in test scans. Our framework for sampling design also involves image reconstruction as one of its components. While the proposed sampling design algorithm is general, we illustrate its usefulness here using the synthesis dictionary-based adaptive reconstruction framework proposed in our prior work [5] (i.e., Problem (P5.0) in Chapter 5). We plan to combine the proposed scheme with the transform-based BCS reconstruction scheme of Chapter 5 as part of future work.

The rest of this chapter is organized as follows. Section 6.2 discusses our framework for adaptive sampling pattern design. In Section 6.3, we present experimental results demonstrating the usefulness of the proposed algorithm for compressed sensing MRI. Finally, in Section 6.4, we conclude.

6.2 Design of the Sampling Pattern

6.2.1 Optimization Problem

We work with a fully sampled training image scan(s) for this section. The goal is to choose a sampling pattern in k-space that gives the best reconstruction of the training image(s) at a given undersampling factor (M). The entire k-space of the training image(s) is partitioned into J *cells*. Examples of such cells for Cartesian and 2D pseudo random sampling are shown in Figure 6.1. The total number of sample points is kept fixed, but they can be redistributed by moving a sample point from one cell to another. The cost function that we optimize is

$$\text{(P6.1)} \quad \min_{S_M} \max_{j} \sum_{i=1}^{N} \left\| \left(y_j^i - H_j \left(S_M y^i \right) \right) \odot G_j \right\|_2^2$$

where y^i represents the k-space values of the i^{th} reference image (N references are assumed), S_M is the undersampling mask in k-space at the undersampling factor M (i.e., $S_M y^i$ represents the undersampled k-space measurements), H

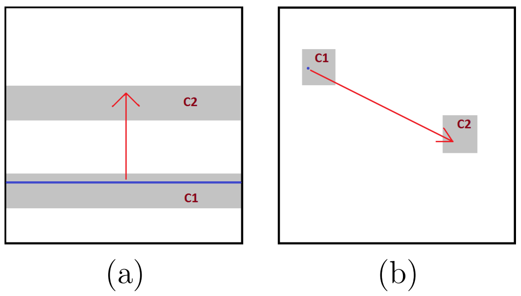


Figure 6.1: Sampling Design: Two cells, C1 and C2, are shown in k-space for (a) Cartesian sampling, and (b) 2D pseudo random sampling. The arrows show the phase encode/sample point in C1 being moved to another location in C2.

represents the reconstruction method for obtaining the full k-space from the undersampled one, and G is a weighting function for k-space. The subscript j is used to index the values in the j^{th} k-space cell C'_j, so that $y^i_j \in \mathbb{C}^{|C'_j|}$, where $|C'_j|$ denotes the number of pixels in C'_j. The problem (P6.1) thus minimizes the maximum (weighted) reconstruction error in the k-space cells for the reference image(s).

Recently, Seeger et al. [147] also proposed optimization of k-space sampling for CSMRI. The optimization there was done sequentially on a single sagittal brain slice using information gain as the criterion, and the resulting sampling pattern was tested on other test data using the reconstruction strategy of Lustig et al. [3]. However, as opposed to their work, we optimize directly the errors in k-space and our cost function is adapted to both the training data and the reconstruction strategy. We also work at higher undersampling factors than [147] and do not a priori dedicate samples for the k-space center.

Our problem formulation (P6.1) of finding the optimal undersampling pattern S_M is combinatorial and NP-hard. We propose an approximate algorithm in Section 6.2.2 for its solution. In order to optimize the cost function in (P6.1), we start with an initial pattern, and iteratively modify it based on the quality of the reconstruction it provides in k-space. A key idea that distinguishes the proposed approach, is that the reconstruction results in each step provide not only a measure of the effectiveness of the sampling pattern used, but are also employed to prescribe *how* the pattern should be modified to improve the results.

6.2.2 Algorithm

Our algorithm alternates between two steps - reconstruction of reference image(s) with a fixed sampling pattern (reconstruction update step), and update of the sampling pattern given the reconstructions (sampling update step). In the following, we will first work with the case $N = 1$ (single reference) for simplicity, and then generalize to the multiple reference/training images case.

The training image is first reconstructed using some specific reconstruction algorithm (e.g., [5], or the new method proposed in Chapter 5) from the initial undersampling pattern. This reconstruction is then transformed to k-space using the discrete Fourier transform (DFT), producing the reconstructed k-space data, which is then partitioned into cells similarly to the training k-space. In each of the cells, we compute the l_2 norm of the difference between the reconstructed and training (original) k-space data. This value is normalized (weighted) by the p-th power of the peak magnitude of the training k-space data in the cell (i.e., $G_j = \frac{1}{\|y_j\|_\infty^p}$ is constant over the k-space locations within the cell [2]), producing the *total cell error.*

The total errors for the various cells are then sorted in increasing order and the cells corresponding to the top L error values are chosen and modified (improved) during the sample re-distribution process (sampling update step). These are the "bad" cells that require more samples to reduce their errors (choosing $L > 1$ allows more than 1 cell to be improved, while also reducing the cost in (P6.1)). Denote the cells sorted in ascending order according to their total errors by $\{C_j\}_{j=1}^J$, with the corresponding total errors $\{E_j\}_{j=1}^J$. Let $e \triangleq sE_{J-L+1}$, where s is a fixed parameter. Thus, e is a measure of the total error in the "best" of the L "bad" cells.

The k-space samples are re-distributed by moving samples from low-error cells to the top L cells. The top L cells are handled sequentially beginning with cell C_J. The non-sampled point (at location P_1) in cell C_J of the reconstructed k-space that has the highest point-wise error is chosen as the candidate to be sampled. The sampled point P_2 from cell C_1 of the reconstructed k-space, that has the lowest point-wise error (note that this error need not be zero as data fidelity may not be exactly enforced during recon-

[2]This is a particular form of the weighting functions G used in this work, and found to work well in our experiments.

struction), is relegated as an ‘unsampled’ point, and a new sample is inserted at location P_1 instead. The value of the reconstructed k-space at P_1 is then set equal to the training value at that point, and at P_2 the value is set to zero. Sample points are sequentially added to C_J at non-sampled locations preferenced by the point-wise reconstruction error (higher errors first). This is done until the total error of C_J falls below e. Furthermore, sample points to be added to C_J are taken from sampled locations of C_1 preferenced by their point-wise reconstruction error (lower errors first). This is done as long as the total error of the reconstructed k-space in C_1 remains below e. Once that condition is violated, sample points are instead chosen for removal from the next cell, i.e. C_2 and so on. Thus, sample points are sequentially moved from the cells with lower total errors to the top L cells (starting with C_J, then C_{J-1} and so on) until the latter have total errors less than e.

It is possible that the total error of a cell (among the top L cells) may fail to diminish below e. This can happen either if we run out of sampled points due to saturation of errors (near e) in all the low error cells or if there are no more unsampled points left in the high error cell in which case the error there is due to the reconstruction method not enforcing ‘exact’ data consistency (at previously sampled points). Once the sample re-distribution process (sampling update step) is complete, the image is reconstructed using the new sampling pattern.

The sampling design iterates over the two steps of reconstruction and sample/phase encode re-distribution. Different values of the power factor p can result in different types of re-distribution (corresponding to different weighting functions G). The value $p = 0$ implies constant weighting (of 1 for all cells) on the k-space reconstruction error, which generally implies that cells (with less samples) near the center of k-space will get higher preference (more likely to be in the top L bad cells) due to higher signal energy concentration near the k-space center. Thus, the predominant movement of sampled points in this case would be towards the center of k-space. Higher values of p result in more general re-distribution of sampled points.

While the algorithm is outlined for a single training scan, it can be easily extended to the case of multiple training images (of same size and scan parameters) by working with cumulative (over the training set) errors of k-space cells (as per the cost in (P6.1)) in the sampling update step (and a suitably defined/chosen weighting functions G).

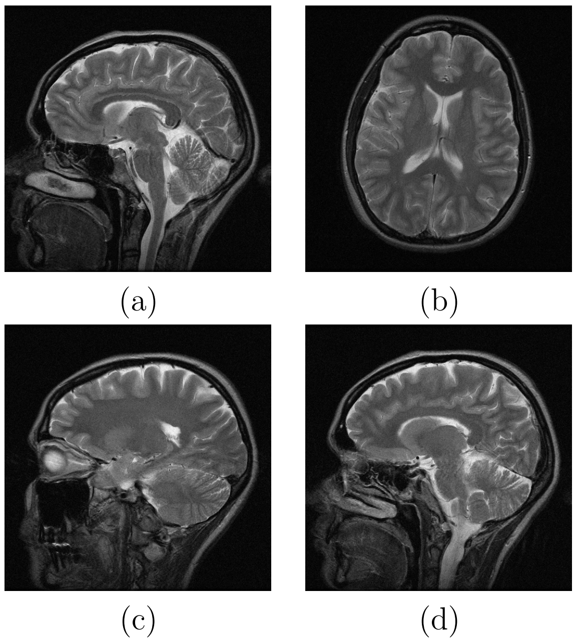


Figure 6.2: Data: (a) Training image, and (b)-(d) test images. Only the magnitudes of the complex-valued images are displayed.

6.3 Numerical Experiments

We performed simulations to test the performance of our sampling framework. The training and test images (512×512 complex MRI scans kindly provided by Prof. Michael Lustig, UC Berkeley) used in our simulations are shown in Fig. 6.2 [3]. The training image was chosen as one slice (with rich features) of a multi-slice data acquisition. In the numerical experiments, we work with simulated k-space data that are obtained by 2D DFT of the complex MR images. The undersampling patterns designed with the training image were tested on other slices of the multi-slice acquisition as well as on a test image from a different scan (Fig. 6.2(b)). 2D pseudo random sampling (Fig. 6.3) and Cartesian sampling (Fig. 6.7 on page 154) schemes are used. For simplicity, we employ here the recent DLMRI algorithm [5] as the reconstruction method H in (P6.1). The parameters of the DLMRI algorithm were set as $n = 36, K = n, T_0 = 7, \lambda = 140$. The k-space of the training slice is shown in Fig. 6.7.

[3]We display only the magnitudes of the complex-valued images in Fig. 6.2. In later Figures (in this chapter), we similarly display only the magnitudes of the complex-valued image reconstructions.

In Fig. 6.3, 5.3-fold undersampling is employed on the k-space of the training slice. The parameters for the sampling design are set as $J = 16384, p = 0.25, s = 0.74, L = 52$, and cell size of 4×4. Five iterations of the algorithm are executed (with no sampling update step in the last iteration) and the initial (variable density random pattern) and final sampling patterns are shown in Fig. 6.3. Based on the locations at which samples were added/removed (Fig. 6.3(c)), it can be inferred that the algorithm captures the underlying k-space structure. The peak signal to noise ratio (PSNR) computed (in dB) as the scaled (by the factor 20) base-10 logarithm of the ratio of the peak intensity value of the original image to the root mean square (rms) reconstruction (from the undersampling pattern) error for that image (rms error computed between image magnitudes) is plotted over algorithm iterations for the training image reconstruction (Fig. 6.3(d)). The PSNR improves by 7 dB with iterations indicating that our sampling design algorithm leads to better sampling patterns. The PSNR also converges quickly indicating fast algorithmic convergence.

The training image reconstructions with the initial and final undersampling patterns shown in Fig. 6.4 depict the improvement in reconstruction quality with adaptive sampling. The reconstruction error magnitudes (computed as $\left| |\widehat{I}| - |I| \right|$, where I is the original image and $\widehat{I}$ is the reconstruction) displayed in Fig. 6.4 also show errors of much smaller magnitude and structure for the final sampling pattern compared to the initial one.

The initial and final undersampling patterns are also tested on the test images. The test reconstructions and error maps shown in Figs. 6.5 and 6.6 show up to 5.5 dB improvement in reconstruction PSNR with the adapted sampling pattern compared to the initial pattern. The significant improvements on a variety of test images indicate the promise of our adaptive sampling design.

Fig. 6.7 employs Cartesian sampling with 4.3-fold undersampling of k-space. The parameters for sampling design are set as $J = 51, p = 0.04, s = 0.74, L = 3$, and cell size of 10 in the phase-encoding direction. Five iterations of the algorithm are executed. The final undersampling pattern (Fig. 6.7(d)) is shown along with the initial pattern (Fig. 6.7(c)). Since $p = 0.04$ is small, sample points move more towards the center of k-space in this case. The PSNR of the training slice reconstruction is plotted over algorithm iterations (Fig. 6.7(b)). It increases by nearly 6.3 dB indicating good improvements

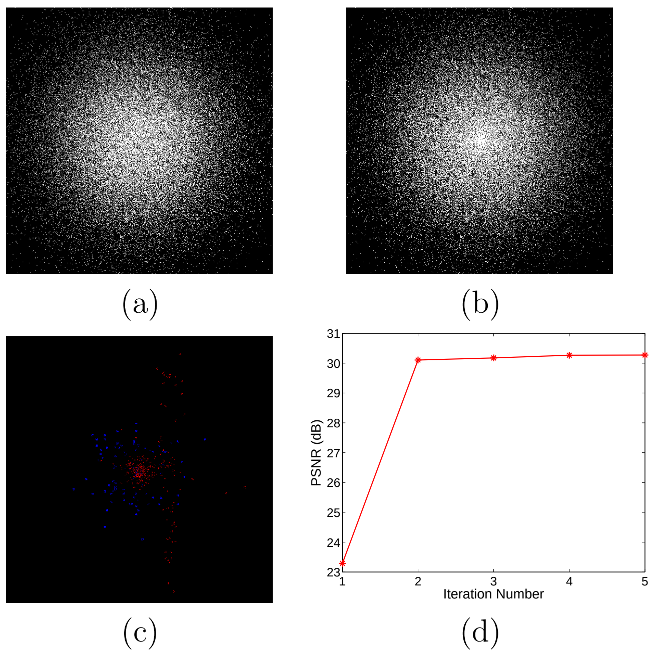


Figure 6.3: 2D random sampling at 5.3-fold undersampling: (a) Initial sampling pattern; (b) final sampling pattern; (c) k-space locations (compare final and initial patterns) where samples were added (in red) and removed (in blue) after 5 iterations; and (d) plot of training image reconstruction PSNR over iterations beginning with initial reconstruction.

on training data. The initial and adapted undersampling patterns are also tested on the test images. The test reconstructions and error maps shown in Fig. 6.8 show up to 6.6 dB improvement in reconstruction PSNR with the adapted sampling pattern compared to the initial one. The promising improvements in reconstruction performance shown on both training and test data indicate the superior performance of adaptive sampling design.

6.4 Conclusions

In this chapter, we introduced an adaptive sampling framework for CSMRI. The iterative algorithm for sampling design utilizes fully sampled training image scans to adapt an initial undersampling pattern. The k-space errors of the image reconstructed from the undersampled k-space data are reduced in each iteration. Significant improvements in reconstruction PSNR were observed in both training and test images when using the adapted sampling

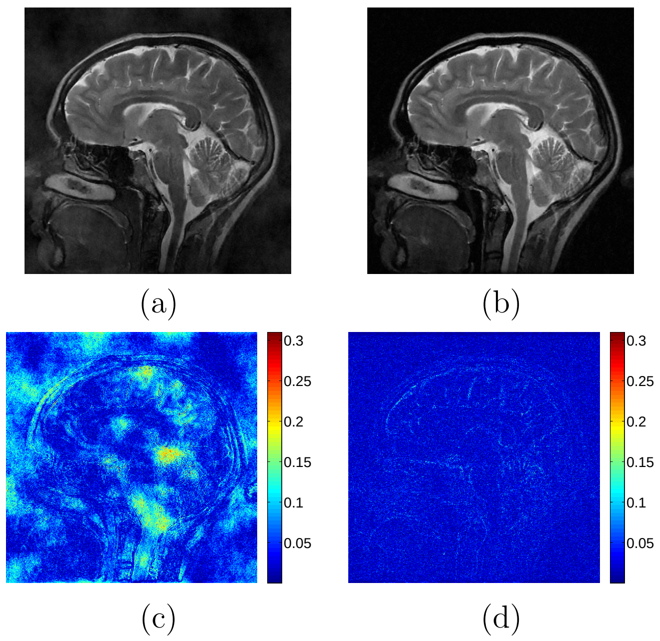


Figure 6.4: Results for training data: Training image reconstruction (image magnitude is displayed) with (a) initial sampling pattern (PSNR = 23.3 dB), and (b) final sampling pattern (PSNR = 30.3 dB). (c)-(d) Reconstruction error magnitudes for (a)-(b).

pattern compared to the initial pattern. The proposed framework for sampling design is generic and can be combined with any reconstruction strategy. We plan to combine it with the transform-based reconstruction method proposed in Chapter 5, in future work. A more detailed study of the parameters involved in sampling design and a comparison to the work of Seeger et al. [147] will be presented elsewhere. We also plan to study the performance of alternative choices for the k-space error weighting function (G).

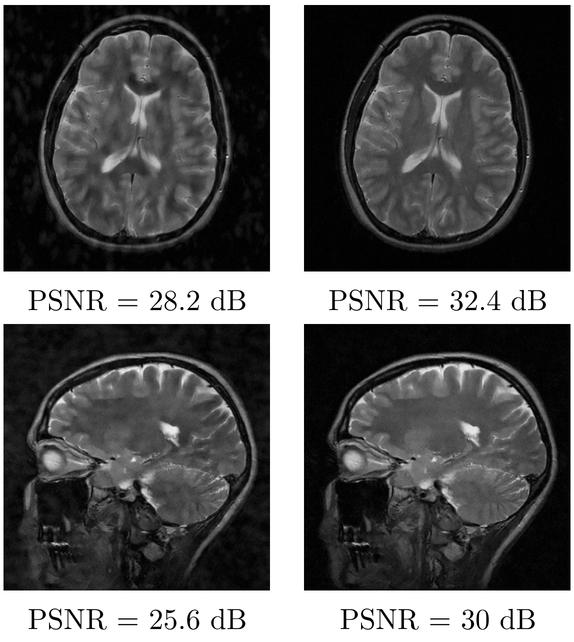


Figure 6.5: Results for test data: Test image reconstructions (image magnitudes are displayed) with initial sampling pattern (left) and final pattern (right).

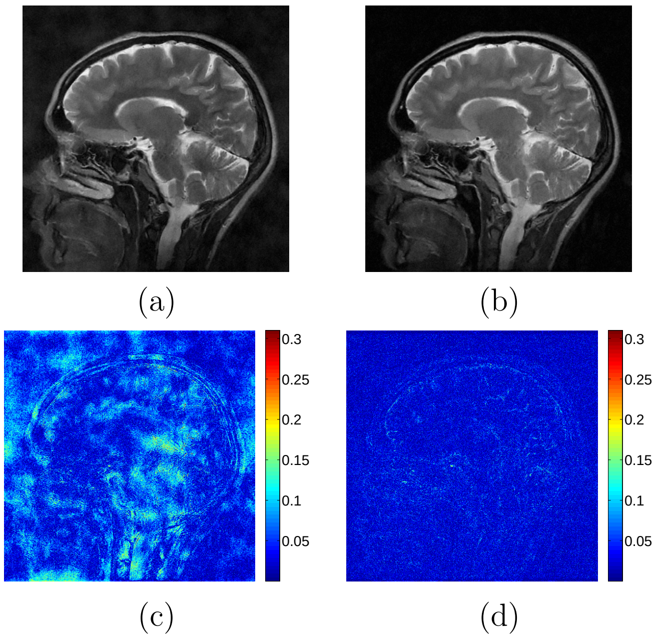
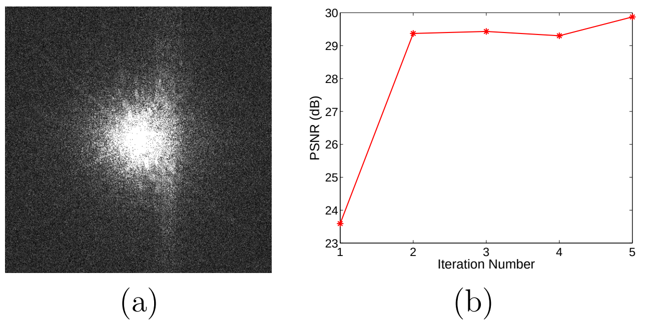
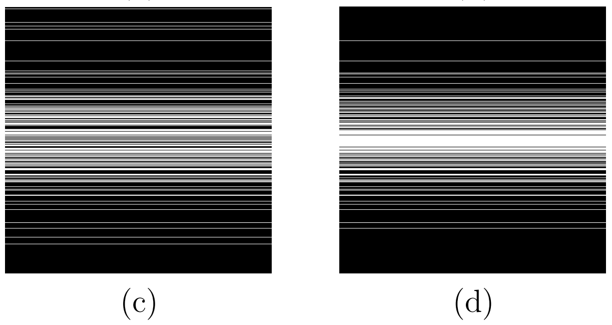


Figure 6.6: Results: Test image reconstruction (image magnitude is displayed) with (a) initial sampling pattern (PSNR = 24.1 dB), and (b) final pattern (PSNR = 29.6 dB). (c)-(d) Reconstruction error magnitudes for (a)-(b).





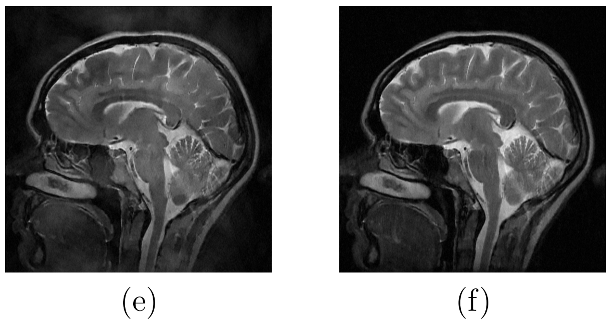


Figure 6.7: Cartesian sampling at 4.3-fold undersampling: (a) Training image k-space; (b) plot of training image reconstruction PSNR over iterations beginning with initial reconstruction; (c) initial sampling pattern; (d) final sampling pattern; Training image reconstruction (image magnitude is displayed) with (e) initial sampling pattern (PSNR = 23.6 dB), and (f) final sampling pattern (PSNR = 29.9 dB).

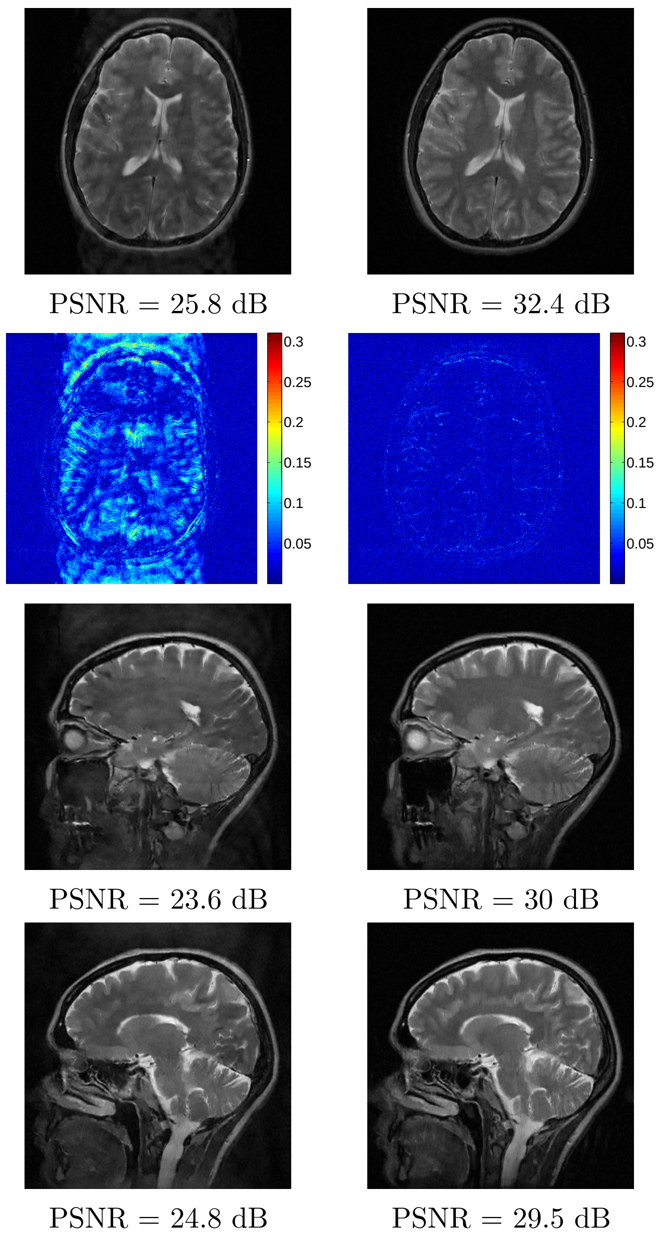


Figure 6.8: Results for test data: Test image reconstructions (image magnitudes are displayed) with initial sampling pattern (left) and final pattern (right). Second row: Reconstruction error magnitudes for first row images.

CHAPTER 7

ONLINE SPARSIFYING TRANSFORM LEARNING - PART I: ALGORITHMS

7.1 Introduction

In the previous chapters, we have shown the learning of transform models to be much cheaper than synthesis or analysis dictionary learning [9, 2]. Adaptive transforms also provide competitive or useful signal reconstruction quality in applications [9, 104, 2, 97, 98]. In this chapter, we develop a methodology for online learning of square sparsifying transforms [1]. Such online learning can be particularly useful when dealing with big data, and for signal processing applications such as real-time sparse representation and denoising. In the following, we introduce the subject of online model learning, and then briefly present the main contributions of this work.

7.1.1 Online Learning and Big Data

Prior work on transform learning focused on batch learning [9, 6], where the sparsifying transform is adapted using all the training data simultaneously. However, for big data, the dimensions of the training data set are typically very large. Hence, batch learning of a sparsifying transform using existing alternating algorithms [6] is computationally expensive in both time and memory, and may be even infeasible. Moreover, in real-time applications, the data arrives sequentially, and must also be processed sequentially to limit latency. Thus, this setting renders batch learning infeasible, since in real-time applications, one does not have access to all the data at once. To address this problem, we introduce in this work a scheme for online, or sequential learning of a square sparsifying transform.

[1]This is a joint work with B. Wen (equal contributor) at the University of Illinois. Parts of the material in this chapter will appear in [148, 149].

Our framework adapts sequentially the sparsifying transform and sparse codes (and/or signal estimates) for signals (or, measurements) that arrive, or are processed sequentially. Such online/sequential transform learning is amenable to big data, and applications such as real-time sparse representation (compression), denoising, and compressed sensing. As we show in this work, online transform learning involves cheap computations and modest memory requirements. Moreover, in Part II (see Chapter 8) of this work [150], we provide strong convergence guarantees for online transform learning. Our numerical experiments illustrate the usefulness of our schemes for big data processing (online sparse representation, and denoising). As we show, the sequential transform learning scheme also converges faster than the batch transform learning scheme [9, 6] in practice.

While the online learning of synthesis dictionaries has been studied previously [20, 19, 151, 152, 153], the online adaptation of the transform model allows for much cheaper computations. Furthermore, the proof by Mairal et al. [20] of the convergence of online synthesis dictionary learning requires various restrictive assumptions. In contrast, our analysis (see Chapter 8 for details) relies on simpler and easily verifiable assumptions. Another feature distinguishing our formulation is that in the previous work, the objective is biconvex, so that the non-convexity in the problem vanishes when a particular variable is kept fixed. This is not the case in our formulation, in which the non-convexity is due to the ℓ_0 "norm" and the log determinant terms. Our formulation remains non-convex even when one of the variables is fixed.

Other very recent works consider synthesis dictionary learning for big data. Wang et al. [154] propose a scheme to incrementally add new columns to the learned dictionary for every new block of signals (sequentially) processed. However, the dictionary size in this method grows continuously (as more blocks of signals are processed), which is undesirable. Another recent work on synthesis dictionary learning for big data is a split and merge learning algorithm [155], which however, is not an online algorithm. The big dataset is split into subsets, and dictionaries are learned in parallel for each subset, before being merged to a single smaller dictionary. However, as the size of the big dataset increases, either the size of each subset increases monotonically, or the final merging step becomes more complex, requiring increasing time and memory. Although faster than conventional dictionary learning, the dictionary learned by this method is worse.

We organize the rest of this chapter as follows. Section 7.2 briefly recalls the prior work on batch transform learning, and then presents our proposed problem formulations for online and mini-batch (that handles blocks of signals sequentially) transform learning and denoising. In Section 7.3, we present efficient algorithms to solve our proposed problem formulations, and discuss our algorithms' computational, latency, and memory advantages. Section 7.4 provides experimental results demonstrating the convergence and computational properties of the proposed schemes. We also show results for sparse representation and denoising. In Section 7.5, we conclude.

7.2 Transform Learning Problem Formulations

7.2.1 Batch Learning

In batch learning, the sparsifying transform is adapted to all the training data simultaneously. Given a matrix $Y \in \mathbb{R}^{n\times N}$, whose columns y_i $(1 \leq i \leq N)$ represent all the training signals, the problem of learning an adaptive square sparsifying transform W in batch mode was formulated in Chapter 4 as follows [9, 6]:

$$\text{(P7.0)} \quad \min_{W,X} \; \|WY - X\|_F^2 + \lambda v(W) \quad s.t. \quad \|x_i\|_0 \leq s \;\; \forall \; i$$

where x_i denotes the i$^{\text{th}}$ column of the sparse code matrix X, s is a given sparsity level, and $v(W) = -\log|\det W| + \|W\|_F^2$. The term $\|WY - X\|_F^2$ in (P7.0) is the sparsification error for the data Y in the transform W, and $v(W)$ is a regularizer to prevent trivial solutions, and control the condition number and scaling of W. To make the two terms in the cost of (P7.0) scale similarly, we set $\lambda = \lambda_0 \|Y\|_F^2$ with constant $\lambda_0 > 0$.

7.2.2 Online Learning

We now introduce our problem formulation for online sparsifying transform learning. The goal here is to adapt the transform and sparse code to data that arrive, or are processed sequentially. For time $t = 1, 2, 3, ...$, the optimization problem to update the sparsifying transform and sparse code based on new data $y_t \in \mathbb{R}^n$ is as follows:

$$\text{(P7.1)} \quad \left\{\hat{W}_t, \hat{x}_t\right\} = \underset{W, x_t}{\arg\min} \ \frac{1}{t} \sum_{j=1}^{t} \left\{ \|Wy_j - x_j\|_2^2 + \lambda_j v(W) \right\}$$
$$s.t. \ \ \|x_t\|_0 \leq s, \ x_j = \hat{x}_j, \ 1 \leq j \leq t-1$$

where $\lambda_j = \lambda_0 \|y_j\|_2^2 \ \forall j$, $\hat{W}_t$ is the optimal transform at time t, and $\hat{x}_t$ is the optimal sparse code for y_t. Note that only the latest sparse code is updated at time t. The condition $x_j = \hat{x}_j, \ 1 \leq j \leq t-1$, is therefore assumed. For brevity, we will not explicitly restate this condition (or, its appropriate variant) in the formulations in the rest of this chapter. On the other hand, at each time t the transform $\hat{W}_t$ is optimized using all the data $\{y_j\}_{j=1}^t$ and sparse codes $\{x_j\}_{j=1}^t$ up to time t. Problem (P7.1) is simply an online version of the batch problem (P7.0), and hence it shares some properties with (P7.0). Specifically, the constant λ_0 controls the condition number of the learned transform.

Although Problem (P7.1) outputs an optimal $\hat{W}_t$ for each t, it is typically impractical to store (in memory) $\hat{W}_t$ for all t. In our experiments, we store only the latest $\hat{W}_t$, and use it as an initialization for the algorithm that solves for $\hat{W}_{t+1}$. At any time instant t, one can obtain a least squares estimate of the signals $\{y_j\}_{j=1}^t$ from their sparse codes as $\left\{\hat{W}_t^{-1} x_j\right\}_{j=1}^t$ (i.e., 'decompressing' the signals from stored sparse codes).

For small values of t, Problem (P7.1) may highly overfit the transform to the data. This is typically undesirable. In order to overcome this problem, for small values of t, we only perform an update of the sparse codes (with a fixed W – set to a reasonable initialization).

Problem (P7.1) can be further modified, or improved in certain scenarios. For example, for non-stationary data, it may not be possible to fit a single transform W to y_t for all t. In this case, one can introduce a forgetting factor ρ^{t-j} (with a constant $0 < \rho < 1$), that scales the terms in (P7.1). Such a forgetting factor would diminish the influence of "old" data. The objective function (within the minimization) in (P7.1) is then modified as

$$\frac{1}{t} \sum_{j=1}^{t} \rho^{t-j} \left\{ \|Wy_j - x_j\|_2^2 + \lambda_j v(W) \right\} \tag{7.1}$$

Note that this is only one form of the forgetting factor (cf. [20] for another form).

For fixed size data sets, Problem (P7.1) can be used as an effective sequential learning and sparse coding (compression) strategy. In this case, it is typically useful to cycle, or make multiple passes through the data set to overcome the causality restriction on the update of the sparse codes. In this case, the same training signals are used, or examined multiple times by (P7.1), which crucially allows to better update the sparse code using a transform determined by the entire data. Similar strategies have been proposed for online synthesis dictionary learning [20].

7.2.3 Mini-batch Learning

A useful variation of online learning is mini-batch learning [20], where we process more than one signal at a time. Mini-batch learning may provide potential reduction in operation count over online learning. However, the processing of blocks of signals leads to increased latency, and memory requirements.

Assuming a fixed block size (or, mini-batch size) of M, the J^{th} ($J \geq 1$) block of signals (in terms of the time sequence $\{y_t\}$) is $Y_J = \left[y_{JM-M+1} \mid y_{JM-M+2} \mid \ \dots \ \mid y_{JM} \right]$. For $J = 1, 2, 3, ...$, the mini-batch sparsifying transform learning problem is formulated as follows:

$$\left\{ \hat{W}_J, \hat{X}_J \right\} = \underset{W, X_J}{\arg\min} \frac{1}{JM} \sum_{j=1}^{J} \left\{ \|WY_j - X_j\|_F^2 + \Lambda_j v(W) \right\}$$
$$s.t. \ \ \|x_{JM-M+i}\|_0 \leq s \ \ \forall\, i \in \{1, .., M\} \quad \text{(P7.2)}$$

where the weight $\Lambda_j = \lambda_0 \|Y_j\|_F^2$, and the matrix $X_J = \left[x_{JM-M+1} \mid x_{JM-M+2} \mid \ \dots \ \mid x_{JM} \right]$ contains the block of sparse codes corresponding to the block Y_J.

Note that both Problems (P7.1) and (P7.2) handle signals sequentially, or involve sequential learning. However, (P7.1) handles one signal at a time, whereas (P7.2) uses blocks of signals at a time. In order to clearly distinguish between these two cases in the rest of this chapter (and in Chapter 8), we will use the terminology 'online learning' to refer to only the case where one signal is processed at a time instant, and we use 'mini-batch learning' to explicitly refer to the case $M > 1$.

7.2.4 Online Denoising Formulation

Online (and mini-batch) transform learning could be used for various applications such as sparse representation (compression), denoising, compressed sensing, etc. Here, we consider an extension of (P7.1) and (P7.2) (which by themselves, can be used for sparse representation of signals) to denoising. Denoising aims to recover an estimate of the signal $z \in \mathbb{R}^n$ from its measurement $y = z + h$, corrupted by noise h. Here, we consider a time sequence of measurements $\{y_t\}$, with $y_t = z_t + h_t$, and $h_t \in \mathbb{R}^n$ being the noise. We assume h_t whose entries are independent identically distributed (i.i.d.) Gaussian with zero mean and variance σ^2. The goal of online denoising is to recover estimates of $z_t \; \forall \; t$. We model the underlying noiseless signals z_t as approximately sparse in a (unknown) transform domain.

Previous work [2, 9] presented a formulation for adaptive sparsifying transform-based batch denoising. Here, we instead present a simple denoising formulation that is a modification of the online learning Problem (P7.1). For $t = 1, 2, 3, ...$, we solve

$$\text{(P7.3)} \quad \min_{W, x_t} \frac{1}{t} \sum_{j=1}^{t} \left\{ \|Wy_j - x_j\|_2^2 + \lambda_j v(W) + \tau_j^2 \, \|x_j\|_0 \right\}$$

where the weights $\tau_j \propto \sigma$. Problem (P7.3) estimates $\hat{W}_t$, $\hat{x}_t$, and the denoised signal is computed simply as $\hat{z}_t = \hat{W}_t^{-1} \hat{x}_t$. Similar to the extension of (P7.1) to (P7.2), we can also extend (P7.3) to its mini-batch version. The different variations of (P7.1) suggested in Section 7.2.2 (such as forgetting factor, and cycling) can also be applied here.

Problem (P7.3) can also be used for patch-based denoising of large images [1, 2], or image sequences. The overlapping patches of the noisy images are processed sequentially, and the denoised image is obtained by averaging the denoised patches at their respective image locations.

7.3 Algorithms and Properties

7.3.1 Batch Transform Learning

In Chapter 4, we proposed an alternating algorithm for solving Problem (P7.0) that alternates between solving for X (*sparse coding step*) and W (*transform update step*), with the other variable kept fixed. The sparse coding solution is computed exactly as $\hat{x}_i = H_s(Wy_i)\ \forall\, i$, where the operator $H_s(\cdot)$ zeros out all but the s coefficients of largest magnitude in a vector. If there is more than one choice for the s coefficients of largest magnitude in a vector z, then we choose $H_s(z)$ as the (thresholded) vector for which the indices of the s largest magnitude elements are the lowest possible. The transform update step too has a closed-form solution – see (4.5) in Chapter 4.

The total cost per iteration (of sparse coding and transform update) of the batch transform learning algorithm scales (assuming $n \ll N$) as $O(Nn^2)$. This is much lower than the per-iteration cost of learning an $n \times K$ overcomplete ($K > n$) synthesis dictionary D using K-SVD [17], which scales (assuming that the synthesis sparsity level $s \propto n$, and $K \propto n$) as $O(Nn^3)$. The (local) memory requirement of batch transform, or dictionary learning scales as $O(Nn)$. This cost becomes prohibitive for large N.

7.3.2 Online Transform Learning

Here, we solve Problem (P7.1) at each time instant t by alternating minimization (similar to (P7.0)).

7.3.2.1 Sparse Coding

In the sparse coding step, we solve (P7.1) for x_t with fixed $W = \hat{W}_{t-1}$ (warm start) as follows:

$$\min_{x_t} \|Wy_t - x_t\|_2^2 \ \ s.t.\ \ \|x_t\|_0 \leq s \tag{7.2}$$

The sparse coding solution is given as $\hat{x}_t = H_s(Wy_t)$, with $H_s(\cdot)$ defined as in Section 7.3.1.

7.3.2.2 Exact Transform Update

In the transform update step, we solve (P7.1) with fixed x_t as

$$\min_{W} \frac{1}{t} \sum_{j=1}^{t} \left\{ \|Wy_j - x_j\|_2^2 + \lambda_j v(W) \right\} \tag{7.3}$$

This problem has a closed-form solution (similar to (4.5)). Let $t^{-1} \sum_{j=1}^{t} \left(y_j y_j^T + \lambda_j I \right) = L_t L_t^T$ (e.g., the positive definite or eigenvalue decomposition (EVD) square root L_t). We compute the full singular value decomposition (SVD) of $L_t^{-1} \Theta_t = Q_t \Sigma_t R_t^T$, where $\Theta_t = t^{-1} \sum_{j=1}^{t} y_j x_j^T$. Then, a closed-form solution [2] to (7.3) is given as

$$\hat{W}_t = 0.5 R_t \left(\Sigma_t + \left(\Sigma_t^2 + 2\beta_t I \right)^{\frac{1}{2}} \right) Q_t^T L_t^{-1} \tag{7.4}$$

where the $(\cdot)^{\frac{1}{2}}$ operation denotes the positive definite square root. We can compute $\Gamma_t \triangleq (t)^{-1} \sum_{j=1}^{t} y_j y_j^T$, Θ_t, and $\beta_t \triangleq \sum_{j=1}^{t} (t)^{-1} \lambda_j$ sequentially over time. However, computing the inverse square root L_t^{-1}, the matrix-matrix product $L_t^{-1} \Theta_t$, and its full SVD would all cost $O(n^3)$ computations. Instead, we propose a computationally cheaper transform update algorithm as follows.

7.3.2.3 Efficient Approximate Transform Update

The following algorithm involves efficient SVD computations, and eliminates matrix-matrix multiplications. To compute the transform update solution, we first efficiently factorize $t^{-1} \sum_{j=1}^{t} \left(y_j y_j^T + \lambda_j I \right)$ as $L_t L_t^T$ (i.e., take square root), with $L_t \in \mathbb{R}^{n \times n}$. Here, we work with the eigenvalue decomposition (EVD) square root. Denoting the full EVD of $\Gamma_{t-1} = (t-1)^{-1} \sum_{j=1}^{t-1} y_j y_j^T$ as $U_{t-1} \Delta_{t-1} U_{t-1}^T$, the full EVD of $\Gamma_t = (1 - t^{-1}) \Gamma_{t-1} + t^{-1} y_t y_t^T$ can be found via a rank-1 update to the (scaled) EVD of Γ_{t-1} [156]. Furthermore, let $\beta_{t-1} = \sum_{j=1}^{t-1} (t-1)^{-1} \lambda_j$. Then, $\beta_t = (1 - t^{-1}) \beta_{t-1} + t^{-1} \lambda_t$. The EVD square root of $t^{-1} \sum_{j=1}^{t} \left(y_j y_j^T + \lambda_j I \right)$ is then computed as $L_t = U_t \left(\Delta_t + \beta_t I \right)^{\frac{1}{2}} U_t^T$ and its inverse as $L_t^{-1} = U_t \left(\Delta_t + \beta_t I \right)^{-\frac{1}{2}} U_t^T$.

The matrix-matrix products in the formula for L_t^{-1} are not explicitly computed. Instead, we will only need the application of L_t^{-1} to a vector, which

[2] The solution (7.4) is unique if and only if $L_t^{-1} \Theta_t$ has full rank.

can be performed efficiently with $O(n^2)$ computation by applying U_t^T, the diagonal matrix $(\Delta_t + \beta_t I)^{-\frac{1}{2}}$, and U_t in succession.

In order to compute the closed-form solution to (7.3), we need to also compute the full SVD of $t^{-1}\sum_{j=1}^{t} L_t^{-1} y_j x_j^T$. In order to simplify this computation, we perform the following approximation:

$$L_t^{-1}\Theta_t = L_t^{-1}\left\{(1-t^{-1})\Theta_{t-1} + t^{-1} y_t x_t^T\right\} \tag{7.5}$$

$$\approx (1-t^{-1})L_{t-1}^{-1}\Theta_{t-1} + t^{-1}L_t^{-1}y_t x_t^T \tag{7.6}$$

With the above approximation, and the fact that $t^{-1}L_t^{-1}y_t x_t^T$ is a rank-1 matrix, the estimate of the full SVD of $t^{-1}\sum_{j=1}^{t} L_t^{-1} y_j x_j^T = L_t^{-1}\Theta_t$ can be obtained by performing a rank-1 update [156] to the scaled (by $1-t^{-1}$) SVD (estimate) of $L_{t-1}^{-1}\Theta_{t-1}$.

Now, once the full SVD estimate of $L_t^{-1}\Theta_t$ is computed as $Q_t\Sigma_t R_t^T$ (compute only the matrices in this decomposition, not the products), the closed-form solution for Problem (7.3) is simply

$$\hat{W}_t = 0.5 R_t \left(\Sigma_t + \left(\Sigma_t^2 + 2\beta_t I\right)^{\frac{1}{2}}\right) Q_t^T L_t^{-1} \tag{7.7}$$

Again, we do not perform any of the matrix-matrix multiplications in (7.7). Instead, we store the individual matrices, and apply them one by one on vectors, at a computational cost of $O(n^2)$.

Note that the only approximation in the above algorithm arises in (7.6). The net error in the approximation in (7.6) at time t (i.e., the difference between $L_t^{-1}\Theta_t$ and its SVD estimate at time t [3]) is given as $E_t = \sum_{j=2}^{t} \frac{j-1}{t}\Upsilon_j$, where $\Upsilon_j = \left(L_j^{-1} - L_{j-1}^{-1}\right)\Theta_{j-1}$. The proof of this result is in Appendix D. In the formula of E_t, the Υ_j for smaller j values gets scaled by smaller numbers (i.e., $(j-1)/t$). It is shown in Part II (Chapter 8) that Υ_j decays (in norm) as C/j, for some constant C. Based on that result, it is easy to show that the approximation error E_t is bounded by C for all t.

In order to prevent any undesirable error accumulations (over time), one may monitor the relative error $\|E_t\|_F / \left\|L_t^{-1}\Theta_t\right\|_F$. The relative error can be shown to be upper bounded (up to a scale factor [4]) by

[3]Note that SVD estimates computed at time j are further used in the rank-1 update at $j+1$.

[4]The factor is $\frac{\max_{2\le j<t}\|\Theta_j\|_2}{\sigma_n(\Theta_t)}$, where σ_n is the smallest singular value of a matrix, and the $\|\cdot\|_2$ norm denotes the spectral norm. The factor is finite for Θ_t that is full rank.

$\sum_{j=2}^{t}(\left\|L_j^{-1} - L_{j-1}^{-1}\right\|_F / \left\|L_t^{-1}\right\|_F)$. The latter quantity is a simplistic upper bound, and is cheap to compute at $O(n^2)$ cost, and can be monitored. If it rises above a small threshold ϵ, we compute the SVD of $L_t^{-1}\Theta_t$ directly, in which case any possible accumulated error is wiped out. In our experiments, we observed that L_t^{-1} converges quickly [5] (over t) for data consisting of natural signals. In such cases, the exact SVD of $L_t^{-1}\Theta_t$ can be obtained for a few initial time instances, after which the approximation (7.6) is observed to work very well. In fact, we observed reasonable performance, even with keeping L_t^{-1} fixed beyond a small t.

An alternative way to perform the transform update (7.3) would be to use the stochastic gradient descent method. However, the gradient computation requires computing the inverse of a matrix (a term like W^{-T} [9]). This computation scales worse ($O(n^3)$) than the computation for the proposed method.

While one could alternate multiple times (for each t) between the sparse coding and transform update steps of (P7.1), we perform only a single alternation to save computations, and to prevent overfitting to the current data. Our overall algorithm for (P7.1) is shown in Fig. 7.1.

7.3.2.4 Handling Variations to (P7.1)

Our algorithm can be easily modified to accommodate the various modifications to Problem (P7.1) suggested in Section 7.2.2 for non-stationary data, and for fixed data sets. For example, when cycling over a fixed data set, the update formula (7.6) would have the term $t^{-1}L_t^{-1}y_t x_t^T$ replaced by $t^{-1}L_t^{-1}y_t(x_t - x_t')^T$, where x_t' is the 'older' version of the sparse code of y_t, which is removed from the formula (and the objective). When the sparse codes are not themselves stored, one can adopt a similar technique as in [20], or use a forgetting factor (7.1) when cycling over the data set, in order to forget the 'older' bad sparse codes.

When using the forgetting factor ρ (as in (7.1)) in online learning, the vari-

[5]For example, when y_t are independent and identically distributed, $t^{-1}\sum_{j=1}^{t} y_j y_j^T$ converges (as $t \to \infty$) with probability 1 to a covariance matrix, and L_t^{-1} would also converge.

[6]If transform update isn't performed for some initial t (Section 7.2.2), then all SVDs are computed exactly for the first transform update. For simplicity, the monitoring of the relative error for (7.6) is not shown in Fig. 7.1.

```
Online Transform Learning Algorithm A1
Input: The sequence $\{y_t\}$.
Initialize: $\hat{W}_0 = W_0$, $\Theta_0 = \Gamma_0 = L_0 = 0$, $\beta_0 = 0$.
For  $t = 1, 2, 3, ...$ Repeat
  1. Sparse Coding: $\hat{x}_t = H_s(\hat{W}_{t-1} y_t)$.
  2. Update $\beta_t = (1 - t^{-1})\beta_{t-1} + t^{-1}\lambda_0 \|y_t\|_2^2$.
  3. Transform Update:
     (a) Compute $(U_t, \Delta_t, U_t^T) \triangleq \mathrm{SVD}(\Gamma_t)$ as the full
         SVD $\left((1 - t^{-1})\Gamma_{t-1} + t^{-1} y_t y_t^T\right)$ by rank-1 update.
     (b) The full $\mathrm{SVD}(L_t^{-1}) = (U_t, (\Delta_t + \beta_t I)^{-\frac{1}{2}}, U_t^T)$.
     (c) Compute $(Q_t, \Sigma_t, R_t^T) \triangleq \mathrm{SVD}(L_t^{-1}\Theta_t)$ as full
         SVD $\left((1 - t^{-1})L_{t-1}^{-1}\Theta_{t-1} + t^{-1} L_t^{-1} y_t x_t^T\right)$ by rank-1 update.
     (d) Store the matrices in the following decomposition
         $\hat{W}_t = 0.5 R_t \left(\Sigma_t + (\Sigma_t^2 + 2\beta_t I)^{\frac{1}{2}}\right) Q_t^T L_t^{-1}$.
End
```

Figure 7.1: Algorithm A1 to solve (P7.1) by alternating minimization [6].

ous operations for the transform update step of (P7.1) are modified as follows. We find the SVD square root L_t of $t^{-1} \sum_{j=1}^{t} \rho^{t-j} \left(y_j y_j^T + \lambda_j I\right)$, and compute the full SVD of $t^{-1} \sum_{j=1}^{t} \rho^{t-j} L_t^{-1} y_j x_j^T$. The methodology of transform update remains the same as before, except that we work with the (modified) matrices/scalars $\Gamma_t = \rho(1 - t^{-1})\Gamma_{t-1} + t^{-1} y_t y_t^T$, $\beta_t = \rho(1 - t^{-1})\beta_{t-1} + t^{-1}\lambda_t$, and $\Theta_t = \rho(1 - t^{-1})\Theta_{t-1} + t^{-1} y_t x_t^T$, in the aforementioned steps.

7.3.2.5 Computational and Memory Costs

We now discuss the computational cost and memory requirements of the online transform learning algorithm. The computational cost of the sparse coding step is dominated [9] by the computation of the product $W y_t$, and therefore scales as $O(n^2)$. In contrast, the projection operation in (7.2) requires only $O(n \log n)$ operations, when employing sorting [9]. The computational cost of the transform update step is dominated by $O(n^2 \log^2 n)$ for the rank-1 SVD updates [156]. Thus, the total cost per signal (or, per time instant) of our algorithm (sparse coding and transform update) scales as $O(n^2 \log^2 n)$. This is better (especially for large n) than the computational cost per signal for online learning of an $n \times K$ overcomplete synthesis dictionary D, which scales (assuming synthesis sparsity $s \propto n$, and $K \propto n$) as at least $O(n^3)$ [20]. The (local) memory requirement of our algorithm scales modestly as $O(n^2)$, since we need to store $n \times n$ matrices.

7.3.3 Mini-batch Transform Learning

Here, we solve Problem (P7.2), that processes blocks of signals, by (a single iteration of) alternating minimization. In the sparse coding step, we solve for X_J in (P7.2), with fixed W $(= \hat{W}_{J-1}$, i.e., warm start) as follows:

$$\min_{X_J} \|WY_J - X_J\|_F^2 \;\; s.t. \;\; \|x_{JM-M+i}\|_0 \leq s \; \forall \, i \tag{7.8}$$

The optimal solution to (7.8) is obtained as $\hat{x}_{JM-M+i} = H_s(Wy_{JM-M+i})$ $\forall$ $i \in \{1, .., M\}$.

The transform update step solves (P7.2) with fixed $\{X_j\}_{j=1}^{J}$ as

$$\min_{W} \frac{1}{JM} \sum_{j=1}^{J} \left\{ \|WY_j - X_j\|_F^2 + \Lambda_j \, Q(W) \right\} \tag{7.9}$$

When the block size M is small ($M \ll n$), we can use the same (approximate) transform update procedure as in Section 7.3.2.3, but with the rank-1 updates replaced by rank-M updates. The rank-M updates can be performed as M rank-1 updates for small M. For larger M ($M \sim O(n)$, or larger), (7.9) is solved using an exact transform update procedure (similar to the one for Problem (7.3)). Fig. 7.2 shows the overall algorithm for the case of larger M.

We now discuss the computational cost and memory requirements of the mini-batch version of online transform learning. The computational cost of the sparse coding step scales as $O(Mn^2)$. For small M, the cost of the transform update step scales as $O(Mn^2 \log^2 n)$. For large M, since the transform update is performed as in Fig. 7.2 (i.e., matrix inverses, matrix-matrix multiplications, and SVDs are computed directly, but scalars/matrices are accumulated over time wherever possible), the cost of transform update scales as $C_1 M n^2 + C_2 n^3$, where C_1 and C_2 are constants. Assuming that $C_2 n < C_1 M$ (large M), the transform update cost scales as $O(Mn^2)$. Thus, the total com-

```
Mini-batch Transform Learning Algorithm A2
Input: Sequence $\{y_t\}$ is processed in blocks of size $M$.
Initialize: $\hat{W}_0 = W_0$, $\tilde{\Theta}_0 = \tilde{\Gamma}_0 = 0$, $\tilde{\beta}_0 = 0$.
For  $J = 1, 2, 3, ...$ Repeat
  1. Sparse Coding: $\hat{x}_l = H_s(\hat{W}_{J-1}\ y_l)\ \forall\ l$ such that $JM - M + 1 \leq l \leq JM$.
  2. Accumulating Various Matrices, or Scalars:
       $\tilde{\Theta}_J = (1 - J^{-1})\,\tilde{\Theta}_{J-1} + J^{-1}M^{-1}Y_J X_J^T.$
       $\tilde{\Gamma}_J = (1 - J^{-1})\,\tilde{\Gamma}_{J-1} + J^{-1}M^{-1}Y_J Y_J^T.$
       $\tilde{\beta}_J = (1 - J^{-1})\,\tilde{\beta}_{J-1} + J^{-1}M^{-1}\lambda_0 \|Y_J\|_F^2$
  3. Transform Update:
     (a) Compute the full SVD of $\tilde{\Gamma}_J + \tilde{\beta}_J I$ as $\tilde{U}_J \tilde{\Delta}_J \tilde{U}_J^T$.
     (b) Obtain $\tilde{L}_J^{-1} = \tilde{U}_J \tilde{\Delta}_J^{-\frac{1}{2}} \tilde{U}_J^T$ (inverse square root).
     (c) Compute full SVD of $\tilde{L}_J^{-1}\tilde{\Theta}_J$ as $\tilde{Q}_J \tilde{\Sigma}_J \tilde{R}_J^T$.
     (d) $\hat{W}_J = 0.5\tilde{R}_J \left(\tilde{\Sigma}_J + \left(\tilde{\Sigma}_J^2 + 2\tilde{\beta}_J I\right)^{\frac{1}{2}}\right) \tilde{Q}_J^T \tilde{L}_J^{-1}$.
End
```

Figure 7.2: Algorithm A2 to solve (P7.2) for large block size M.

putation per iteration (or, per block) of our mini-batch algorithm (sparse coding and transform update) scales as $O(Mn^2)$ for large M, and $O(Mn^2 \log^2 n)$ for small M. In either case, the cost is better than the cost per block (of size M) for mini-batch learning of an $n \times K$ synthesis dictionary D, which scales (assuming synthesis sparsity $s \propto n$, and $K \propto n$) as $O(Mn^3)$ [20]. The memory requirement of mini-batch transform learning scales as $O(Mn)$ for large M, and $O(n^2)$ for small M.

7.3.4 Comparison of Transform Learning Schemes

We now compare and contrast the online, mini-batch, and batch transform learning schemes in terms of their computational costs, memory requirements, and latency. We measure latency as the time duration (the inter-arrival time between two signals is taken as 1 time unit) between the arrival of the first signal, and the generation of its corresponding output (e.g., sparse code) [7].

Table 7.1 summarizes the various costs for the transform-based schemes. We show the computational cost per sample, i.e., the cost normalized by the number of samples processed. For a given number of N samples, the batch scheme typically requires several iterations to converge to a good transform. Thus, the batch scheme for learning a good transform has the total per sample computational cost of $O(PNn^2)$, where P is the total number of batch iterations. In practice P depends on n, N, and algorithm initialization, and it typically becomes larger for bigger, or more complex problems. On the other hand, as shown in Part II (Chapter 8), and in the experiments of Section 7.4, the online and mini-batch schemes produce good transforms for N (total number of signals processed sequentially) large. Therefore, the net computational cost for processing N signals (and converging) is $O(Nn^2 \log^2 n)$. Thus, assuming $\log^2 n < P$ (which is typically observed in practice), the online scheme is computationally more effective (in order) than the batch scheme for big data. The computational cost of processing N signals (and thus converging, in the case of large N) for the mini-batch scheme is $O(Nn^2)$ (with large M and N/M blocks), which is even lower in order (by factor $\log^2 n$) than that for the (one signal at a time) online scheme.

[7]Here, for simplicity, we assume that computations can be performed instantaneously.

Properties	Online	Mini-batch		Batch
		Small M	Large M	
Computations	$O(n^2 \log^2 n)$	$O(n^2 \log^2 n)$	$O(n^2)$	$O(Pn^2)$
Memory	$O(n^2)$	$O(n^2)$	$O(nM)$	$O(nN)$
Latency	0	$M-1$	$M-1$	$N-1$

Table 7.1: Comparison of online learning, mini-batch learning, and batch learning in terms of their computational cost per sample, memory cost, and latency.

Importantly, assuming $n, M \ll N$, the online and mini-batch schemes have far lower memory requirements and latency compared to the batch scheme. The mini-batch scheme itself has higher memory and latency costs than the online scheme.

As discussed in the preceding subsections, the dictionary learning schemes [20, 17] have a computational cost per sample (not shown in Table 7.1) proportional to n^3. For large signals (i.e., large n), the dictionary update cost is more prohibitive than the transform update cost.

7.3.5 Denoising

Problem (P7.3) is identical to (P7.1), except for the fact that it uses a sparsity penalty function, rather than constraints. Therefore, when solving (P7.3) at each t by alternating minimization, the sparse coding step (with fixed $W = \hat{W}_{t-1}$) is

$$\min_{x_t} \|Wy_t - x_t\|_2^2 + \tau_t^2 \|x_t\|_0 \tag{7.10}$$

A solution $\hat{x}_t$ of (7.10) is $\hat{x}_t = \hat{H}_{\tau_t}(Wy_t)$, where the hard thresholding operator $\hat{H}_\tau(\cdot)$ is defined as

$$\left(\hat{H}_\tau(b)\right)_k = \begin{cases} 0 & , \ |b_k| < \tau \\ b_k & , \ |b_k| \geq \tau \end{cases} \tag{7.11}$$

where $b \in \mathbb{R}^n$, and the subscript k indexes vector entries. Therefore, the sparse coding solution is simply obtained by hard thresholding, with a threshold proportional to the noise level σ (similar to traditional techniques involving analytical transforms [157]). The transform update step of (P7.3) is identical to (P7.1). The denoised signal is computed as $\hat{W}_t^{-1}\hat{x}_t$. By, (7.7), we have

$$\hat{W}_t^{-1} = \beta_t^{-1} L_t Q_t \left((\Sigma_t^2 + 2\beta_t I)^{\frac{1}{2}} - \Sigma_t \right) R_t^T \tag{7.12}$$

Assuming that the various matrices in the above decomposition are stored in memory, $\hat{W}_t^{-1}\hat{x}_t$ can be computed using matrix-vector multiplications at a cost of $O(n^2)$. The computational cost of denoising per signal then scales as $O(n^2 \log^2 n)$ (as in (P7.1)). For mini-batch based denoising, the computational cost of denoising a block is similar to the costs mentioned in Section 7.3.3.

7.4 Numerical Experiments

7.4.1 Framework

In Chapter 8 (Part II of this work), we prove that online transform learning converges asymptotically, and produces a good transform. Here, we present numerical results illustrating the practical convergence behavior of online (and mini-batch) transform learning, as well as the usefulness of the proposed schemes for image representation and denoising. First, we consider synthetic data generated sequentially using a particular transform model, and study the ability of our online schemes to converge to a good model. Second, we briefly study the usefulness of online/sequential learning for sparse representation of images. Finally, we present results for online denoising using Problem (P7.3). We consider the patch-based denoising of some standard (regular sized) images as well as some very large images (where batch learning was observed to be infeasible on the particular computing platform used for our experiment). The latter case is a candidate big data problem, since it involves a large number of patches, which can be potentially denoised efficiently and sequentially using online transform learning.

All transform learning implementations were coded in Matlab version R2013b. Similarly, the Matlab implementation of K-SVD denoising [1] (a popular batch synthesis dictionary-based denoising scheme) available from Michael Elad's website [86] was used in our comparisons. For K-SVD denoising, we used the built-in parameter settings of the author's implementation. All computations were performed with an Intel Core i7 CPU at 2.9GHz and

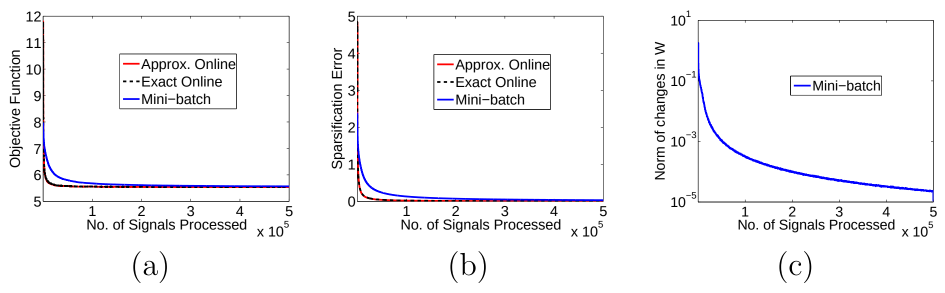


Figure 7.3: Convergence behavior of the online (both the exact and approximate versions) and mini-batch learning schemes as a function of amount of data processed: (a) Objective function, (b) Sparsification error, (c) $||\hat{W}_{t+1} - \hat{W}_t||_F$ for mini-batch scheme.

4GB memory, employing a 64-bit Windows 7 operating system.

We define the normalized reconstruction error as $\|Y - W^{-1}X\|_F^2 \,/\, \|Y\|_F^2$, where Y is a matrix of data vectors, W is a transform, and X is the corresponding sparse code matrix. The normalized reconstruction error metric is used to measure the performance of learned transforms for signal/image representation. It can be thought of as a simple surrogate for the compression performance of a transform. For image denoising, similar to prior work, we measure the peak-signal-to-noise ratio (PSNR) computed between the true noiseless reference, and the noisy or denoised images.

7.4.2 Convergence and Sparse Representation

7.4.2.1 Convergence

First, we illustrate the convergence behavior of the proposed algorithms. We generate the input data y_t sequentially as $W^{-1}x_t$ using a random unitary 20×20 matrix W, and sparse codes x_t obtained by thresholding i.i.d. Gaussian vectors at sparsity level $s = 3$. We then use our online and mini-batch transform learning algorithms for (P7.1) and (P7.2), to sequentially learn the transform and sparse codes for the data y_t. The parameter $\lambda_0 = 3.1 \times 10^{-2}$, $s = 3$, and the size of the mini-batch $M = 320$. We test (and compare) both the exact (see Section 7.3.2.2) and approximate (see Section 7.3.2.3) versions of the online transform learning algorithm. As discussed in Section 7.3.2.3, we monitor the upper bound on the relative approximation error. If it rises above a threshold $\epsilon = 0.1$, we compute the SVD of $L_t^{-1}\Theta_t$ directly in the transform update step of the algorithm. Our algorithms are initialized with the 20×20 DCT matrix.

Figs. 7.3(a) and 7.3(b) show the objective function (of Problems (P7.1)/(P7.2)) and sparsification error (i.e., the objective without the regularizer) as a function of the amount of signals processed, for our online and mini-batch schemes. Both the objective and sparsification error converge quickly for our schemes. The exact and approximate online schemes behave identically. Moreover, for the approximate version, we observed that the error threshold ϵ is violated (i.e., an exact SVD is performed) only 0.025% of the time. This indicates that the faster approximate online scheme works equally well as the exact version. The online schemes converge slightly faster than the mini-batch scheme as a function of the number of signals processed. This is because the transform update step is performed more frequently for the (one signal at a time) online schemes. However, the online schemes (typically) have a higher run time (due to the $\log^2 n$ factor in computations – see Section 7.3.4) than the mini-batch scheme.

As shown in Fig. 7.3(c), the difference between successive iterates, i.e., $\left\|\hat{W}_{t+1} - \hat{W}_t\right\|_F$ converges close to zero for the mini-batch scheme. A similar behavior is observed for the online scheme. The learned transforms using our exact online, approximate online, and mini-batch algorithms have condition numbers as 1.02, 1.02, and 1.04 respectively. By Fig. 7.3(b), they provide a sparsification error close to zero. The normalized reconstruction error computed using the sparse codes generated sequentially is < 0.01 for our schemes, indicating that they have learned a good model for the data $\{y_t\}$.

7.4.2.2 Sparse Representation of Images

We have demonstrated the potential of the proposed online and mini-batch transform learning schemes for sparse representation of image patches in [148]. Specifically, a large set of patches (5×10^5 patches of size 8×8) were extracted from the images in the USC-SIPI database [158] (the color images were converted to gray-scale images). The patches were extracted from random locations in the images. We used our online and mini-batch transform learning algorithms to learn a transform and sparse codes ($s = 11$) sequen-

Images	σ	Noisy PSNR		Batch K-SVD	Batch TL	Mini-batch TL
Couple	5	34.16	PSNR	37.28	37.33	37.33
			time	1250	92	20
	10	28.11	PSNR	33.51	33.62	33.62
			time	671	68	19
	20	22.11	PSNR	30.02	30.02	30.03
			time	190	61	20
Man	5	34.15	PSNR	36.47	36.66	36.75
			time	1279	205	45
	10	28.13	PSNR	32.71	32.96	33.00
			time	701	130	44
	20	22.11	PSNR	29.40	29.57	29.52
			time	189	80	41

Table 7.2: PSNR values (dB) and run times (seconds) for denoising small images at different noise levels ($\sigma = 5$, 10, 20), using batch K-SVD [1], batch transform learning [6], and our mini-batch transform learning-based denoising.

tially on the (mean-subtracted) patches. The normalized reconstruction error was used to quantify the patch representation quality. It was shown that both the online and mini-batch schemes (either with or without cycling [8]) provided better reconstruction quality compared to fixed transforms such as the DCT. The proposed schemes with a few cycles (5 cycles), were also shown to perform similarly as the batch transform learning scheme. Importantly, the mini-batch scheme was shown to be much faster ($6\times$ faster in the experiment of [148]) than the batch algorithm in achieving similar reconstruction quality.

7.4.3 Online Image Denoising

7.4.3.1 Regular-size Image Denoising

Here, we present some results for our simple denoising framework (P7.3). We consider image denoising, where the overlapping image patches are processed and denoised sequentially. We work with the images Couple (512×512) and

[8]By cycling, we mean that we make multiple passes through the same data (each time a particular signal is repeated, its old sparse code is replaced with the latest one – see Section 7.3.2.4).

Man (768×768) [9], and simulate i.i.d. Gaussian noise at three different noise levels ($\sigma = 5$, 10, 20) for the images.

We denoise the 8×8 overlapping image patches (the mean is subtracted during learning, and added back in the reconstruction step) sequentially (no cycling) using adaptive mini-batch denoising. Once the denoised patches are computed (block by block), we immediately put them back at their corresponding locations in the denoised image. Note that the denoised image is computed by averaging the denoised patches at their respective 2D locations. Our scheme requires minimal memory (we only store data required for computations at a particular time instant t) and mimics a real-time denoising setup.

We work with a forgetting factor (cf. (7.1)) in our formulation [10]. The parameter is set to $\lambda_0 = 3.1 \times 10^{-2}$, $M = 64$, and $\tau_j = 1.73\sigma$. The forgetting factor is set to $\rho = 0.87$, 0.95, and 0.99, for $\sigma = 5$, 10, and 20, respectively. These values were found empirically [11].

Our denoising results are compared to K-SVD denoising [17, 1, 86], and batch square transform (parameters set as in [2]) denoising [6]. The images in this experiment have sizes compatible (i.e., not large enough to create memory overflows) with the batch denoising schemes. The goal of the comparison to the batch dictionary/transform schemes is not to show the state-of-the-art performance of our method in a general denoising application. Rather, we focus on the sequential aspect of our method, and aim to demonstrate that the proposed adaptive mini-batch transform denoising algorithm with less latency, memory and computational requirements, can be used as an efficient and effective alternative to adaptive batch mode dictionary/transform denoising.

Table 7.2 lists the denoising PSNRs and run times (including the time for final denoised image generation) for the various methods. The mini-batch transform denoising method provides comparable, or better denoising performance compared to the batch-based methods, while being much faster. We compute the average speedup provided by our mini-batch denoising scheme over the adaptive batch-based methods. For each image and noise level, the

[9]These are standard images that have been used in prior work (e.g., [1, 2]).

[10]Results without a forgetting factor are presented in [149]. We observed slightly better denoising with a forgetting factor.

[11]Typically, the forgetting factor ρ (that works best) depends on the size of the mini-batch, patch size (signal size), and noise level.

ratio of the run times of batch denoising and mini-batch denoising is computed, and this speedup is averaged over the images and noise levels in Table 7.2. The mini-batch scheme provides an average speedup of 26.0× and 3.4× respectively, over the batch K-SVD and batch transform denoising schemes.

7.4.3.2 Large-Scale Image Denoising

In the context of big data, batch learning is typically infeasible due to the strict practical limits on computational and memory resources. However, we can potentially use the proposed online, or mini-batch transform learning schemes to sparse code, or denoise large images, and image sequences. In [148], we have presented results for mini-batch denoising of large images (both gray-scale and color images). The largest of those images had about 11 Megapixels (when processed by a patch-based scheme, there are about 11 million overlapping patches in total for such an image). For several large images and noise levels, we showed in [148] that the mini-batch denoising algorithm provides much better PSNRs than fixed transforms such as the DCT. Importantly, despite the fact that there is no learning involved in the latter case, the adaptive scheme typically denoised about as fast as the fixed transform.

As we have emphasized, the proposed adaptive online and mini-batch schemes are capable of being applied to realistic tasks such as real-time sparse coding (compression), and denoising. The idea of image inpainting using mini-batch synthesis dictionary learning has been discussed in [20]. However, the scheme therein does not solve a real-time adaptive inpainting problem. Rather, a dictionary is first learned from the uncorrupted data, and then later used to reconstruct (with fixed dictionary) the corrupted patches. Therefore, we do not directly compare to that work here.

7.5 Conclusions

In this chapter, we presented a novel problem formulation for online learning of square sparsifying transforms. The formulation is to sequentially update the sparsifying transform and sparse code for signals that arrive or, are processed sequentially. The proposed algorithm involves a sparse coding step and a transform update step per signal. Each of these steps is implemented efficiently. We also presented a mini-batch version of our online algorithm that can handle blocks of signals at a time. The proposed schemes were shown to be computationally much cheaper (in terms of cost per signal) than online synthesis dictionary learning. In practice, the online/mini-batch sparsifying transform learning converges better/faster than batch mode (where all signals are considered simultaneously) transform learning. We presented experiments demonstrating the usefulness of online transform learning in sparse signal representation, and denoising. The topics of online learning of an overcomplete transform, and online video denoising will be considered in future work.

CHAPTER 8

ONLINE SPARSIFYING TRANSFORM LEARNING - PART II: CONVERGENCE ANALYSIS

8.1 Introduction

This chapter, the theoretical counterpart to our work in Chapter 7 on data-driven online learning of sparsifying transforms, provides a convergence analysis of the proposed algorithms [1]. We prove that although the associated optimization problems are highly non-convex, our online transform learning algorithms in Chapter 7 are guaranteed to converge to the set of stationary points of the learning problem. The guarantee relies on few (easy to verify) assumptions. In practice, our alternating algorithms work well, as demonstrated by sample applications to representing and denoising signals in Chapter 7.

While the online learning of synthesis dictionaries has been studied or analyzed previously [20, 19, 151, 152, 153], the online adaptation of the transform model allows for much cheaper computations [148]. Furthermore, the proof by Mairal et al. [20] of the convergence of online synthesis dictionary learning requires various restrictive (see Section 8.2 for details) assumptions. In contrast, our analysis relies on simpler and easily verifiable assumptions. Another feature distinguishing our formulation is that in the previous work, the objective is biconvex, so that the non-convexity in the problem vanishes when a particular variable is kept fixed. This is not the case in our formulation (e.g., Problem (P7.1)), in which the non-convexity is due to the ℓ_0 "norm" and the log determinant terms. Our formulation remains non-convex even when one of the variables (either the transform, or the sparse code) is fixed.

The rest of this chapter is devoted to the convergence analysis of the algorithms in Chapter 7. We will mostly focus on the convergence behavior of the algorithm that solves Problem (P7.1), and briefly mention corresponding

[1]The material of this chapter will appear in [150].

results for our (similar) algorithm for the block-based (P7.2). The organization of the rest of the chapter is as follows. In Section 8.2, we first present some notations and assumptions for our convergence analysis. Section 8.3 presents the main convergence results. The proof of convergence is detailed in Section 8.4. Finally, in Section 8.5, we conclude.

8.2 Notations and Assumptions

8.2.1 Notations

The objective in Problem (P7.1) at time t is denoted as

$$\tilde{g}_t(W, x_t) \triangleq \frac{1}{t}\sum_{j=1}^{t}\left\{\|Wy_j - x_j\|_2^2 + \lambda_j v(W)\right\} \tag{8.1}$$

where $\{x_j = \hat{x}_j\}_{j<t}$ have been computed at previous t values. Our algorithm for (P7.1) finds the sparse code as $\hat{x}_t = H_s(Wy_t)$, with $W = \hat{W}_{t-1}$. This is followed by a transform update step. Let us denote the objective of the transform update step as

$$\hat{g}_t(W) \triangleq \tilde{g}_t(W, \hat{x}_t) \tag{8.2}$$

For a signal y, transform W, and vector x, we define

$$\tilde{u}(y, W, x) \triangleq \|Wy - x\|_2^2 + \lambda_0 \|y\|_2^2 \, v(W) \tag{8.3}$$

Then, we define the signal-wise loss function $u(y, W)$ as

$$\begin{aligned} u(y, W) &\triangleq \min_{x:\|x\|_0 \le s} \tilde{u}(y, W, x) \\ &= \|Wy - H_s(Wy)\|_2^2 + \lambda_0 \|y\|_2^2 \, v(W) \end{aligned} \tag{8.4}$$

Thus, $u(y, W)$ is small for signals (assuming signals of similar scaling) that are sparsified well by W. We use the operation $\tilde{H}_s(b)$ to denote the *set* of all optimal projections of $b \in \mathbb{R}^n$ onto the s-ℓ_0 ball defined as $\{x \in \mathbb{R}^n : \|x\|_0 \le s\}$. When $\tilde{H}_s(b)$ is a unique element, it satisfies $\tilde{H}_s(b) = H_s(b)$, for $H_s(\cdot)$ defined as in Chapter 7.

We also define the empirical objective function

$$g_t(W) \triangleq \frac{1}{t}\sum_{j=1}^{t} u(y_j, W) \tag{8.5}$$

The empirical objective function involves the optimal sparse code (in W) for each y_j, and it is the objective that is minimized by batch transform learning algorithms [6, 9]. Note however that in an online setting, the sparse codes of past signals cannot be optimally set at future times t.

For convenience, we split the various functions ($\tilde{g}_t(W, x_t)$, $\hat{g}_t(W)$, $u(y, W)$, $g_t(W)$) that have to do with the objective into the sum of two terms: the first, with a superscript of (1) will be used to denote the sparsification error term; the second will denote the regularizer term. For example, $u(y, W) = u^{(1)}(y, W) + u^{(2)}(y, W)$, where $u^{(1)}(y, W) = \|Wy - H_s(Wy)\|_2^2$ and $u^{(2)}(y, W) = \lambda_0 \|y\|_2^2 v(W)$.

8.2.2 Assumptions

In order to derive the convergence results, we will make the following few assumptions.

(A1) Signal Normalization. First, we assume that the input signals y_t are normalized, i.e., $\|y_t\|_2 = 1$ [2]. This assumption eliminates the dependence on y of the regularizer weighting, for the various functions in Section 8.2.1.

(A2) Exact Computation. The transform update step of our algorithm(s) is assumed to be performed exactly (referred to as "exact", since there is a simple closed-form solution involving the SVD [3]). This is always the case for the mini-batch algorithm in the larger M ($M \sim O(n)$) case (cf. Chapter 7). For the algorithm for (P7.1), the exact transform update method in Section 7.3.2.2 is slower than the approximate one in Section 7.3.2.3 [4] of Chapter 7, and has an $O(n^3)$ computational cost per signal, but will be still assumed to be the one used, for the purpose of theoretical analysis.

[2]Any input that is 0 can always be dropped, or processed trivially.

[3]Although in practice the SVD is computed using iterative methods, the methods are guaranteed to quickly provide machine precision accuracy.

[4]As mentioned in Chapter 7, the approximate transform update method performs equally well (as the exact one) in practice. The convergence results in this work may therefore be also relevant to the approximate method.

(A3) Nondegenerate SVD. We assume (for each t) that $L_t^{-1}\Theta_t = t^{-1}\sum_{j=1}^{t} L_t^{-1} y_j x_j^T$ has non-degenerate (distinct, non-zero) singular values, i.e., there is a minimum separation $\hat{\eta} > 0$ between any two singular values as well as between the smallest singular value and zero. We observed this assumption to hold in our experiments. One could also simply monitor the singular values of $L_t^{-1}\Theta_t$ (over t), and drop (i.e., ignore from the formulation/algorithm) the signals y_t for those time instances (t), when the assumption is violated. Such signals could be treated as "outliers" and processed separately [5]. Assumption A3 is not required for showing the convergence of the objective function $\hat{g}_t$ in our algorithms.

(A4) Random Signals. The signals y_t are assumed to be independently and identically distributed over the unit sphere $\{y \in \mathbb{R}^n : \|y\|_2 = 1\}$, according to an absolutely continuous probability density function $p(y)$.

Our assumptions are less restrictive (and also easier to verify) than the ones in [20]. There, the authors assume the uniqueness of the synthesis sparse coding solution. However, such a uniqueness assumption may not hold in general. Moreover, the proof in [20] assumes that the synthesis sparse coding problem is solved exactly at each t (a similar assumption is also made for the dictionary update step in [20]), which is typically impractical [6] in general. Another assumption in [20] is the positive definiteness of the Hessian of the dictionary learning objective (this is similar to our assumption on $L_t^{-1}\Theta_t$). We would also like to emphasize that as opposed to the prior work [20], we work with an optimization problem that is not simply biconvex. Specifically, our problem involves the ℓ_0 "norm" for sparsity, and a non-convex log-determinant penalty.

8.2.3 Expected Transform Learning Cost

Given the statistical assumptions about the signals, we follow the standard approach in the analysis of online algorithms (cf. [159, 160, 161, 162, 20])

[5]Assuming that such outliers occur infrequently, they could for example be processed (sparse coded) using a fixed analytical sparsifying transform such as Wavelets.

[6]In general, the iterative optimization algorithms [20] (for either synthesis sparse coding, or dictionary update) may take a large number of iterations to reach machine precision accuracy. Moreover, these algorithms do not provide a method to determine the accuracy of the computed solution at any given iteration.

and consider the minimization of the expected cost

$$g(W) \triangleq \mathbb{E}_y \left[u(y, W) \right] \tag{8.6}$$

where the expectation is with respect to the (unknown) probability distribution $p(y)$ of the data. It follows from Assumption A4 that $\lim_{t\to\infty} g_t(W) = g(W)$ a.s. (almost sure convergence). In particular, given a specific training set, it may be unnecessary to minimize the empirical objective function $g_t(W)$ to high precision, since it is only an approximation to the expected cost. In fact, even an inaccurate minimizer of $g_t(W)$ could (potentially) provide the same, or better value of the expected cost than a fully optimized one. Although we cannot directly minimize the expected cost, we will show interesting asymptotic properties for our algorithm with respect to the expected cost.

8.3 Main Results

The main convergence results in this work are briefly stated as follows. We assume some particular (non-singular) initialization $\hat{W}_0$ for our algorithms. For simplicity, we state results for the online algorithm for (P7.1). Similar results can be easily shown to hold for the mini-batch scheme. For the sequence $\left\{\hat{W}_t\right\}$ generated by our online scheme, we have

(i) As $t \to \infty$, $\hat{g}_t(\hat{W}_t)$, $g_t(\hat{W}_t)$, and $g(\hat{W}_t)$ converge almost surely to a common limit, say g^*.

(ii) The sequence $\left\{\hat{W}_t\right\}$ is bounded. Every accumulation point $\hat{W}_\infty$ of $\left\{\hat{W}_t\right\}$ is a stationary point of the expected cost $g(W)$ satisfying $\nabla g(\hat{W}_\infty) = 0$, with probability 1.

(iii) Every accumulation point of $\left\{\hat{W}_t\right\}$ achieves the same value (i.e., g^*) of the expected cost g with probability 1.

Statement (i) above shows convergence of the objective function sequences. It is interesting that $\hat{g}_t(\hat{W}_t)$ and $g_t(\hat{W}_t)$ converge almost surely to the same limit. Since the definition of $g_t(\hat{W}_t)$ involves the optimal sparse codes computed using the common $\hat{W}_t$ for all signals y_j $1 \leq j \leq t$, whereas $\hat{g}_t$ uses the sequentially computed $\hat{W}_{j-1}$ for (sparse coding) signal y_j ($1 \leq j \leq t$), the convergence result (i) means that we do not lose (asymptotically) by sparse coding the signals only sequentially.

Statement (ii) above says that every accumulation point of the iterate sequence is a stationary point of the expected cost $g(W)$ with probability 1. Furthermore, Statement (iii) shows that every accumulation point $\hat{W}_\infty$ of $\left\{\hat{W}_t\right\}$ satisfies $g(\hat{W}_\infty) = g^*$ with probability 1. In other words, every accumulation point is equally good in terms of its expected cost (i.e., $g(\cdot)$) value.

We also have the folowing Statement (iv).

(iv) The distance between $\hat{W}_t$ and the set of stationary points of the expected cost $g(W)$ converges to 0 almost surely as $t \to \infty$.

Statement (iv) indicates that the iterate sequence $\left\{\hat{W}_t\right\}$ converges to the set of stationary points of $g(W)$ with probability 1.

Finally, we also show the following other interesting results.

(v) $\hat{g}_{t+1}(\hat{W}_{t+1}) - \hat{g}_t(\hat{W}_t)$ decays as $O(1/t)$.

(vi) $\hat{g}_{t+1}(\hat{W}_{t+1}) - \hat{g}_{t+1}(\hat{W}_t)$ decays as $O(1/t^2)$.

(vii) $\hat{W}_{t+1} - \hat{W}_t$ also decays (in norm) as $O(1/t)$.

The above results show the convergence rate of the difference between successive iterates or objective values. Statement (vi) indicates that the objective decreases at a $O(1/t^2)$ rate within the transform update step of the algorithm. However, when the sparse coding step is included in the calculation (i.e., statement (v) above), the rate of decrease is only $O(1/t)$, due to the uncertainty introduced by a newly added signal. Note that the above statements do not by themselves indicate convergence of the objective, or iterate sequences, but will be used to prove such convergence.

8.4 Proof of Convergence

We now prove the convergence properties of our online algorithm for (P7.1). The various results proved here (leading up to our main convergence results) are listed as follows.

(i) The iterate sequence $\left\{\hat{x}_t, \hat{W}_t\right\}$ generated by our algorithm is bounded.

(ii) The objective sequence $\left\{\hat{g}_t(\hat{W}_t)\right\}$ is also bounded.

(iii) The objective and iterate sequences each have at least one convergent subsequence.

(iv) $\hat{g}_{t+1}(\hat{W}_{t+1}) - \hat{g}_t(\hat{W}_t)$ decays as $O(1/t)$.

(v) $\hat{W}_{t+1} - \hat{W}_t$ also decays (in norm) as $O(1/t)$.

(vi) $\hat{g}_{t+1}(\hat{W}_{t+1}) - \hat{g}_{t+1}(\hat{W}_t)$ decays as $O(1/t^2)$.

(vii) As $t \to \infty$, $\hat{g}_t(\hat{W}_t)$, $g_t(\hat{W}_t)$, and $g(\hat{W}_t)$ converge almost surely to a common limit.

(viii) Each accumulation point of $\left\{\hat{W}_t\right\}$ is a stationary point of the expected cost $g(W)$, with probability 1. Moreover, every accumulation point achieves the same value (g^*) of the expected cost g with probability 1.

(ix) The distance between $\hat{W}_t$ and the set of stationary points of the expected cost $g(W)$ converges to 0 almost surely as $t \to \infty$.

Result (i) above is given by the following lemma. For the simplicity of our proofs, we work with a weighting $\lambda_0 \geq 2$ in this section. This condition leads to a simple bound of unity for the norms of the iterates in our proofs. We can permit smaller values of λ_0 by scaling the y_t's accordingly, so that the upper bound of unity on the norms of the iterates holds for any particular choice of $\lambda_0 < 2$.

Lemma 2. *Under Assumptions A1 and A2, for any* $\lambda_0 \geq 2$*, the iterate sequence* $\left\{\hat{x}_t, \hat{W}_t\right\}$ *generated by our algorithm is bounded* $\forall\ t$ *as* $\left\|\hat{x}_t\right\|_2 \leq 1$*, and* $\left\|\hat{W}_t\right\|_2 \leq 1$*. Furthermore, we have that*

$$\left\|\Sigma_t\right\|_2 \leq \left\|L_t^{-1}\right\|_2 \leq \frac{1}{\sqrt{\lambda_0}}\ \forall\, t \tag{8.7}$$

Proof: Assuming without loss of generality that the initialization is scaled so that $\left\|\hat{W}_0\right\|_2 \leq 1$, we have for $t = 1$

$$\left\|\hat{x}_1\right\|_2 = \left\|H_s(\hat{W}_0 y_1)\right\|_2 \leq \left\|\hat{W}_0 y_1\right\|_2 \leq \left\|\hat{W}_0\right\|_2 \left\|y_1\right\|_2 = 1 \tag{8.8}$$

Furthermore, $\hat{W}_t = 0.5R_t\left(\Sigma_t + (\Sigma_t^2 + 2\beta_t I)^{1/2}\right)Q_t^T L_t^{-1}$ for any t, where $Q_t\Sigma_t R_t^T$ is the full SVD of $L_t^{-1}\Theta_t$ with $\Theta_t = (1/t)\sum_{j=1}^t y_j\hat{x}_j^T$, and $\beta_t = \sum_{j=1}^t (t)^{-1}\lambda_j$. Therefore,

$$\left\|\hat{W}_1\right\|_2 \leq 0.5\left\|\Sigma_1 + (\Sigma_1^2 + 2\beta_1 I)^{\frac{1}{2}}\right\|_2 \left\|L_1^{-1}\right\|_2 \tag{8.9}$$

by the sub-multiplicativity of the matrix spectral norm. Since, $\|y_t\|_2 = 1$ $\forall$ t, we get $\beta_t = \lambda_0$ for all t. Moreover, for every t,

$$\left\|L_t^{-1}\right\|_2 = \left\|\left(\frac{\sum_{j=1}^t y_j y_j^T}{t} + \lambda_0 I\right)^{-1/2}\right\|_2 \leq \frac{1}{\sqrt{\lambda_0}} \tag{8.10}$$

We also have

$$\left\|\Sigma_t\right\|_2 = \left\|L_t^{-1}\Theta_t\right\|_2 \leq \left\|L_t^{-1}\right\|_2 \left(\frac{1}{t}\sum_{j=1}^t \left\|y_j\right\|_2 \left\|\hat{x}_j\right\|_2\right) \tag{8.11}$$

It is then obvious using (8.8) that $\left\|\Sigma_1\right\|_2 \leq \left\|L_1^{-1}\right\|_2$. Substituting this into (8.9), we easily get

$$\left\|\hat{W}_1\right\|_2 \leq \frac{0.5}{\lambda_0} + 0.5\left(\frac{1}{\lambda_0^2} + 2\right)^{1/2} \tag{8.12}$$

It follows that $\left\|\hat{W}_1\right\|_2 \leq 1$, whenever $\lambda_0 \geq 2$ holds. Then, upon repeating the aforementioned arguments for $t = 2, 3$, etc. (equivalently, by induction), we obtain that $\hat{x}_t$ and $\hat{W}_t$ are bounded (same bound of unity as above) for each and every t. Then, (8.7) also holds for every t, just as shown above for the $t = 1$ case. ■

Next, we show that the objective sequence is bounded.

Lemma 3. *Under Assumptions A1 and A2, the objective sequence* $\left\{\hat{g}_t(\hat{W}_t)\right\}$ *is bounded.*

Proof: The transform $\hat{W}_t$ is a minimizer of the transform update objective for fixed $x_t = \hat{x}_t = H_s(\hat{W}_{t-1}y_t)$ (i.e., $\hat{W}_t$ minimizes $\hat{g}_t(W)$). Therefore, we have

$$\hat{g}_t(\hat{W}_t) \leq \hat{g}_t(\hat{W}_0) = \lambda_0 v(\hat{W}_0) + \frac{1}{t}\sum_{j=1}^t \left\|\hat{W}_0 y_j - x_j\right\|_2^2 \tag{8.13}$$

By Lemma 2, we know that the x_j's above (with $x_j = \hat{x}_j$ fixed) are all bounded. Therefore, $\hat{g}_t(\hat{W}_t)$ is also upper bounded. We also have (by results in Chapter 4) that $\hat{g}_t(\hat{W}_t) \geq 0$. Therefore, $\hat{g}_t(\hat{W}_t)$ is bounded for each t. ■

Proposition 8. *The objective and iterate sequences in our algorithm, each have at least one convergent subsequence.*

Proof: Since the objective and the iterate sequences are bounded, the existence of a convergent subsequence (for a bounded sequence) is a standard result. ■

The next two propositions show the $O(1/t)$ decay of the difference between successive elements of the objective and iterate sequences.

Proposition 9. *Let Assumptions A1 and A2 hold. Then, the objective sequence* $\left\{\hat{g}_t(\hat{W}_t)\right\}$ *satisfies*

$$\left|\hat{g}_{t+1}(\hat{W}_{t+1}) - \hat{g}_t(\hat{W}_t)\right| \leq \frac{C}{t+1} \tag{8.14}$$

where $C > 0$ *is a constant independent of* t.

Proof: First, we have by the definition of $\hat{g}_{t+1}$ (8.2) that

$$\hat{g}_{t+1}(\hat{W}_{t+1}) = \frac{t\hat{g}_t(\hat{W}_{t+1})}{t+1} + \frac{\tilde{u}(y_{t+1}, \hat{W}_{t+1}, \hat{x}_{t+1})}{t+1} \tag{8.15}$$

Since $\hat{g}_t(\hat{W}_{t+1}) \geq \hat{g}_t(\hat{W}_t)$, and $\tilde{u}(y_{t+1}, \hat{W}_{t+1}, \hat{x}_{t+1}) \geq 0$, we get

$$\hat{g}_{t+1}(\hat{W}_{t+1}) - \hat{g}_t(\hat{W}_t) \geq -\frac{\hat{g}_t(\hat{W}_t)}{t+1} \geq -\frac{C}{t+1} \tag{8.16}$$

where the last inequality above follows from Lemma 3.

Second, since $\hat{g}_{t+1}(\hat{W}_{t+1}) \leq \hat{g}_{t+1}(\hat{W}_t)$, we also have that $\hat{g}_{t+1}(\hat{W}_{t+1}) - \hat{g}_t(\hat{W}_t) \leq \hat{g}_{t+1}(\hat{W}_t) - \hat{g}_t(\hat{W}_t)$. Combining this with (8.15) (with $\hat{W}_{t+1}$ replaced by $\hat{W}_t$ in (8.15)), we get

$$\hat{g}_{t+1}(\hat{W}_{t+1}) - \hat{g}_t(\hat{W}_t) \leq -\frac{\hat{g}_t(\hat{W}_t)}{t+1} + \frac{\tilde{u}(y_{t+1}, \hat{W}_t, \hat{x}_{t+1})}{t+1} \tag{8.17}$$

Since $-\hat{g}_t(\hat{W}_t) + \tilde{u}(y_{t+1}, \hat{W}_t, \hat{x}_{t+1}) \leq \left\| \hat{W}_t y_{t+1} - \hat{x}_{t+1} \right\|_2^2$ (by algebraic manipulations), we have

$$\hat{g}_{t+1}(\hat{W}_{t+1}) - \hat{g}_t(\hat{W}_t) \leq \frac{\left\| \hat{W}_t y_{t+1} - \hat{x}_{t+1} \right\|_2^2}{t+1} \leq \frac{C}{t+1} \tag{8.18}$$

where the last inequality follows by Lemma 2 and Assumption A1. Combining (8.18) and (8.16), we have the desired result. ■

Proposition 10. *Let Assumptions A1-A3 hold. Then, the sequence* $\left\{ \hat{W}_t \right\}$ *satisfies*

$$\left\| \hat{W}_{t+1} - \hat{W}_t \right\|_F \leq \frac{C}{t} \quad \forall\, t \tag{8.19}$$

where C is a constant independent of t.

Proof: Let $A_t \triangleq 0.5 R_t \left(\Sigma_t + (\Sigma_t^2 + 2\beta_t I)^{1/2} \right) Q_t^T$, where $Q_t \Sigma_t R_t^T$ is the full SVD of $L_t^{-1}\Theta_t$ (cf. Section 7.3.2.2 in Chapter 7), and $\beta_t = \lambda_0$. Then, we have $\hat{W}_{t+1} - \hat{W}_t = (A_{t+1} - A_t)\, L_t^{-1} + A_{t+1} \left(L_{t+1}^{-1} - L_t^{-1} \right)$. Therefore,

$$\begin{aligned} \left\| \hat{W}_{t+1} - \hat{W}_t \right\|_F \leq & \left\| L_t^{-1} \right\|_2 \left\| A_{t+1} - A_t \right\|_F \\ & + \left\| A_{t+1} \right\|_2 \left\| L_{t+1}^{-1} - L_t^{-1} \right\|_F \end{aligned} \tag{8.20}$$

By Lemma 2, $\left\| L_t^{-1} \right\|_2 \leq 1/\sqrt{\lambda_0}$ and $\left\| A_{t+1} \right\|_2 \leq \frac{0.5}{\sqrt{\lambda_0}} + 0.5 \left(\frac{1}{\lambda_0} + 2\lambda_0 \right)^{1/2}$.

We now bound $\left\| L_{t+1}^{-1} - L_t^{-1} \right\|_F$ in (8.20). Defining $K_t \triangleq \frac{\sum_{j=1}^t y_j y_j^T}{t} + \lambda_0 I$, we have that $L_t^{-1} = K_t^{-1/2}$. First, it is easy to show that

$$\| K_{t+1} - K_t \|_F \leq \frac{2}{t+1} \tag{8.21}$$

i.e., $K_{t+1} - K_t = O(1/t)$. Second, using Taylor series expansions of the matrix inverse [7] and square root [8], it can be shown that for large t, $K_{t+1}^{-1/2} = K_t^{-1/2} + E_t$, with $E_t = O(1/t)$. Since the result holds for all sufficiently large (determined by λ_0) t, we can always find a constant c_0 such that $\left\| L_{t+1}^{-1} - L_t^{-1} \right\|_F \leq c_0/t$, $\forall\, t$. Alternatively, we can drop a finite number of sequence elements, so that the result holds for all remaining t.

[7] Assuming $A \in \mathbb{R}^{n \times n}$ is invertible, and B is small, $(A+B)^{-1} = A^{-1} - A^{-1}BA^{-1} + A^{-1}BA^{-1}BA^{-1} - A^{-1}BA^{-1}BA^{-1}BA^{-1} + \ldots$

[8] For small $B \in \mathbb{R}^{n \times n}$, $(I+B)^{1/2} = I + \frac{1}{2}B - \frac{1}{8}B^2 + \frac{1}{16}B^3 - \ldots$

Next, we bound $\left\|A_{t+1} - A_t\right\|_F$ in (8.20). Since A_t is a function of the SVD of $L_t^{-1}\Theta_t$, we first need a bound on

$$\left\|L_{t+1}^{-1}\Theta_{t+1} - L_t^{-1}\Theta_t\right\|_F \leq \left\|L_{t+1}^{-1}\right\|_2 \left\|\Theta_{t+1} - \Theta_t\right\|_F + \left\|\Theta_t\right\|_2 \left\|L_{t+1}^{-1} - L_t^{-1}\right\|_F \tag{8.22}$$

Since $\|\Theta_{t+1} - \Theta_t\|_F \leq \frac{2}{t+1}$, and $\left\|\Theta_t\right\|_2 \leq 1$ (by Lemma 2 and Assumption A1), it is clear that $\left\|L_{t+1}^{-1}\Theta_{t+1} - L_t^{-1}\Theta_t\right\|_F \leq c_1/t$ (with $c_1 = c_0 + (2/\sqrt{\lambda_0})$). Now, we apply Theorem 10 from Appendix E.3 here to conclude that the singular values of $L_{t+1}^{-1}\Theta_{t+1}$ differ from the corresponding singular values of $L_t^{-1}\Theta_t$ only by $O(1/t)$. Moreover, each left and right singular vector pair of $L_{t+1}^{-1}\Theta_{t+1}$ differs (under our Assumption A3 that $L_t^{-1}\Theta_t$ has non-degenerate singular values $\forall\ t$) from the corresponding pair of $L_t^{-1}\Theta_t$ only by [9] $O(1/t)$. Using these perturbation bounds for the singular values and singular vectors (and a simple scalar Taylor series expansion for the diagonal elements of $\left(\Sigma_{t+1}^2 + 2\lambda_0 I\right)^{1/2}$), it is easy to show that $\left\|A_{t+1} - A_t\right\|_F \leq c_2/t$, with c_2 a constant independent of t.

Finally, substituting the bounds on $\left\|A_{t+1} - A_t\right\|_F$ and $\left\|L_{t+1}^{-1} - L_t^{-1}\right\|_F$ into (8.20), equation (8.19) follows. ■

As mentioned before, the fact that the difference between successive iterates, or objective values decays as $O(1/t)$, although interesting, does not by itself indicate convergence of the respective sequences. In order to prove such convergence, we also need the following two Lemmas. The first lemma shows that the objective decreases as $O(1/t^2)$ within the transform update step at time t.

Lemma 4. *Let Assumptions A1 and A2 hold. Then, there exists a constant $c > 0$ independent of t such that*

$$\left|\hat{g}_{t+1}(\hat{W}_{t+1}) - \hat{g}_{t+1}(\hat{W}_t)\right| \leq \frac{c}{t^2} \tag{8.23}$$

[9]A particular pair may be scaled by -1 for the result to hold.

Proof: First, due to the optimality condition (in transform update step) for $\hat{W}_{t+1}$, we have $\hat{g}_{t+1}(\hat{W}_{t+1}) - \hat{g}_{t+1}(\hat{W}_t) \leq 0$. Second, we have that

$$\begin{aligned} \hat{g}_{t+1}(\hat{W}_{t+1}) - \hat{g}_{t+1}(\hat{W}_t) &= \hat{g}_{t+1}(\hat{W}_{t+1}) - \hat{g}_t(\hat{W}_{t+1}) \\ &+ \hat{g}_t(\hat{W}_{t+1}) - \hat{g}_t(\hat{W}_t) + \hat{g}_t(\hat{W}_t) - \hat{g}_{t+1}(\hat{W}_t) \end{aligned} \tag{8.24}$$

Since $\hat{g}_t(\hat{W}_{t+1}) - \hat{g}_t(\hat{W}_t) \geq 0$, we get

$$\begin{aligned} \hat{g}_{t+1}(\hat{W}_{t+1}) - \hat{g}_{t+1}(\hat{W}_t) \geq \hat{g}_{t+1}(\hat{W}_{t+1}) - \hat{g}_t(\hat{W}_{t+1}) \\ + \hat{g}_t(\hat{W}_t) - \hat{g}_{t+1}(\hat{W}_t) \end{aligned} \tag{8.25}$$

We now consider the function $\hat{g}_{t+1} - \hat{g}_t$. This function does not depend on the transform learning regularizer $v(W)$ (it cancels out). It is in fact a quadratic in W. Importantly, since the transforms in our case belong to a bounded set (by Lemma 2), it is easy to show that the function $\hat{g}_{t+1} - \hat{g}_t$ is Lipschitz with respect to W. with a Lipschitz constant $\leq \frac{C_1}{t+1}$. Using this fact in (8.25), we get that

$$0 \geq \hat{g}_{t+1}(\hat{W}_{t+1}) - \hat{g}_{t+1}(\hat{W}_t) \geq \frac{-C_1}{t+1} \left\| \hat{W}_{t+1} - \hat{W}_t \right\|_F$$

Combining the above result with the result (8.19) of Proposition 10, the required condition (8.23) follows. ■

The almost sure convergence of the objective sequence is provided by the following proposition.

Proposition 11. *Let Assumptions A1-A4 hold. Then, as $t \to \infty$, $\hat{g}_t(\hat{W}_t)$, $g_t(\hat{W}_t)$, and $g(\hat{W}_t)$ all converge almost surely to a common limit g^*.*

Proof: We adopt a similar proof methodology here as in [161, 20], but differ in the details, and required conditions/assumptions. Define $r_t = \hat{g}_t(\hat{W}_t)$. Then, $r_t \geq 0$. We will use Theorem 12 in Appendix E.3 to show that r_t is a quasi-martingale and converges almost surely. To apply Theorem 12, we first need to investigate $r_{t+1} - r_t$, which is given as

$$\begin{aligned} r_{t+1} - r_t &= \hat{g}_{t+1}(\hat{W}_{t+1}) - \hat{g}_{t+1}(\hat{W}_t) + \hat{g}_{t+1}(\hat{W}_t) - \hat{g}_t(\hat{W}_t) \\ &\leq \frac{t\hat{g}_t(\hat{W}_t)}{t+1} + \frac{u(y_{t+1}, \hat{W}_t)}{t+1} - \hat{g}_t(\hat{W}_t) \qquad (8.26) \\ &= \frac{g_t(\hat{W}_t) - \hat{g}_t(\hat{W}_t)}{t+1} + \frac{u(y_{t+1}, \hat{W}_t) - g_t(\hat{W}_t)}{t+1} \qquad (8.27) \\ &\leq \frac{u(y_{t+1}, \hat{W}_t) - g_t(\hat{W}_t)}{t+1} \qquad (8.28) \end{aligned}$$

where $\hat{g}_{t+1}(\hat{W}_{t+1}) \leq \hat{g}_{t+1}(\hat{W}_t)$ is used to arrive at (8.26), and $g_t(\hat{W}_t) \leq \hat{g}_t(\hat{W}_t)$ is used to arrive at (8.28). Now, we consider the filtration $\mathcal{F}_t$ (cf. Theorem 12) determined by the past information up to time t as follows

$$\mathbb{E}[r_{t+1} - r_t | \mathcal{F}_t] \leq \frac{\mathbb{E}[u(y_{t+1}, \hat{W}_t)|\mathcal{F}_t] - g_t(\hat{W}_t)}{t+1} \qquad (8.29)$$

$$= \frac{g(\hat{W}_t) - g_t(\hat{W}_t)}{t+1} = \frac{g^{(1)}(\hat{W}_t) - g_t^{(1)}(\hat{W}_t)}{t+1} \leq \frac{\left\|g^{(1)} - g_t^{(1)}\right\|_\infty}{t+1}$$

where $\left\|g^{(1)} - g_t^{(1)}\right\|_\infty = \sup_{W \in S} \left|g^{(1)}(W) - g_t^{(1)}(W)\right|$, with the set S as defined in Lemma 38 of Appendix E.1, and $g^{(1)}(W) = \mathbb{E}_y \left[u^{(1)}(y, W)\right]$.

In order to satisfy the requirements of Theorem 12 (Appendix E.3), we will first bound $\mathbb{E}\left[\sqrt{t}\left\|g^{(1)} - g_t^{(1)}\right\|_\infty\right]$, for which we use Theorem 11 in Appendix E.3. Note that by Lemma 38 in Appendix E.1 $u^{(1)}(y, W)$ is Lipschitz with respect to W on the bounded set S. Moreover, $u^{(1)}(y, W)$ is bounded and $\mathbb{E}_y\left[\left\{u^{(1)}(y, W)\right\}^2\right]$ is also (uniformly) bounded. Therefore, directly applying Theorem 11, we conclude that $\mathbb{E}\left[\sqrt{t}\left\|g^{(1)} - g_t^{(1)}\right\|_\infty\right] = O(1)$. Thus, defining δ_t as in Theorem 12 (Appendix E.3), we get

$$\mathbb{E}\left[\delta_t(r_{t+1} - r_t)\right] = \mathbb{E}\left[\mathbb{E}\left[r_{t+1} - r_t | \mathcal{F}_t\right]^+\right] \leq \frac{c}{t^{\frac{3}{2}}} \qquad (8.30)$$

where c is a constant, and the $(\cdot)^+$ operation zeros out negative numbers. Equation (8.30) immediately implies that the requirement $\sum_{t=1}^{\infty} \mathbb{E}[\delta_t(r_{t+1} - r_t)] < \infty$ is met for Theorem 12 (Appendix E.3). Therefore, as $t \to \infty$, r_t converges almost surely.

The rest of the results are simple to derive. We briefly mention the steps here for completeness. We will now prove that $g(\hat{W}_t)$ converges almost surely.

First, we have from Theorem 12 (Appendix E.3) that

$$\sum_{t=1}^{\infty} \Big|\mathbb{E}[r_{t+1} - r_t|\mathcal{F}_t]\Big| < \infty \text{ a.s.} \tag{8.31}$$

We now use the fact that

$$r_{t+1} - r_t = \hat{g}_{t+1}(\hat{W}_{t+1}) - \hat{g}_{t+1}(\hat{W}_t) + \frac{u(y_{t+1}, \hat{W}_t) - \hat{g}_t(\hat{W}_t)}{t+1}$$

to get the following result

$$\begin{aligned}\sum_{t=1}^{\infty} \frac{\Big|g(\hat{W}_t) - \hat{g}_t(\hat{W}_t)\Big|}{t+1} &\leq \sum_{t=1}^{\infty} \Big|\mathbb{E}[r_{t+1} - r_t|\mathcal{F}_t]\Big| \\ &+ \sum_{t=1}^{\infty} \Big|\mathbb{E}\Big[\hat{g}_{t+1}(\hat{W}_{t+1}) - \hat{g}_{t+1}(\hat{W}_t)|\mathcal{F}_t\Big]\Big| \end{aligned} \tag{8.32}$$

We know from Lemma 4 that

$$\sum_{t=1}^{\infty} \mathbb{E}\Big[\Big|\hat{g}_{t+1}(\hat{W}_{t+1}) - \hat{g}_{t+1}(\hat{W}_t)\Big|\,|\mathcal{F}_t\Big] \leq \sum_{t=1}^{\infty} \frac{c}{t^2} < \infty$$

Using this result and (8.31) in (8.32), we immediately get the almost sure convergence of the following sum

$$\sum_{t=1}^{\infty} \frac{\Big|g(\hat{W}_t) - \hat{g}_t(\hat{W}_t)\Big|}{t+1} = \sum_{t=1}^{\infty} \frac{\Big|g^{(1)}(\hat{W}_t) - \hat{g}_t^{(1)}(\hat{W}_t)\Big|}{t+1} \tag{8.33}$$

We now use (8.33), and apply Theorem 13 in Appendix E.3 to show that $g(\hat{W}_t) - \hat{g}_t(\hat{W}_t) \to 0$ almost surely. Let $b_t = \frac{1}{t+1}$, $a_t = g(\hat{W}_t) - \hat{g}_t(\hat{W}_t)$ $= g^{(1)}(\hat{W}_t) - \hat{g}_t^{(1)}(\hat{W}_t)$, and $d_t = |a_t|$ (using similar notations as in Theorem 13 of Appendix E.3). Then, it is easy to show that

$$\begin{aligned}\Big|d_{t+1} - d_t\Big| \leq \Big|a_{t+1} - a_t\Big| &\leq \Big|\hat{g}_t^{(1)}(\hat{W}_{t+1}) - \hat{g}_t^{(1)}(\hat{W}_t)\Big| \\ &+ \Big|g^{(1)}(\hat{W}_{t+1}) - g^{(1)}(\hat{W}_t)\Big| + \frac{\nu}{t+1}\end{aligned} \tag{8.34}$$

where ν is a constant that upper bounds $\Big|\hat{g}_t^{(1)}(\hat{W}_{t+1})\Big|$ $+ \Big|\tilde{u}^{(1)}(y_{t+1}, \hat{W}_{t+1}, \hat{x}_{t+1})\Big|$. It is also easy to see that both $\hat{g}_t^{(1)}(W)$ (a quadratic) and $g^{(1)}(W)$ (follows using Lemma 38 here) are Lipschitz over W in a compact set. Combining this with the result of Proposition 10, we can easily see that (8.34) implies $\left|d_{t+1} - d_t\right| < \hat{c}/(t+1)$ for some $\hat{c} > 0$ that does not depend on t. Now all the conditions for Theorem 13 in Appendix E.3 are satisfied (i.e., $b_t \geq 0$, $d_t \geq 0$, $\sum_{t=1}^{\infty} b_t d_t < \infty$, $\sum_{t=1}^{\infty} b_t = \infty$, and $|d_{t+1} - d_t| < \hat{c}\, b_t \;\forall\; t$), thereby guaranteeing that $\lim_{t\to\infty} d_t = 0$ almost surely, i.e.,

$$g(\hat{W}_t) - \hat{g}_t(\hat{W}_t) \to 0 \;\; a.s. \tag{8.35}$$

Since we have already proved that $\hat{g}_t(\hat{W}_t)$ converges almost surely, (8.35) implies the almost sure convergence of $g(\hat{W}_t)$.

We also have by the generalized Glivenko-Cantelli Theorem (for example, see Theorem 19.4 in [163]) that $\left\|g_t - g\right\|_{\infty} \to 0$ almost surely as $t \to \infty$. Combining this with the almost sure convergence of $g(\hat{W}_t)$, we finally obtain that $g_t(\hat{W}_t)$ converges almost surely as well. Moreover, with probability 1, the limits of $g(\hat{W}_t)$, $\hat{g}_t(\hat{W}_t)$ and $g_t(\hat{W}_t)$ are identical. ■

The fact that $g(\hat{W}_t) - \hat{g}_t(\hat{W}_t) \to 0$ almost surely (in the above proof) also directly implies that $g^{(1)}(\hat{W}_t) - \hat{g}_t^{(1)}(\hat{W}_t) \to 0$ almost surely. Our final result discusses the convergence of the iterates.

Proposition 12. *Let Assumptions A1-A4 hold. Then, every accumulation point of $\left\{\hat{W}_t\right\}$ is a stationary point of the expected cost $g(W)$ with probability 1. Moreover, every accumulation point achieves the same value g^* of the expected cost with probability 1.*

Proof: Consider a matrix $B \in \mathbb{R}^{n\times n}$. By the definition of g_t, we have that

$$\hat{g}_t(\hat{W}_t + B) \geq g_t(\hat{W}_t + B) \;\forall\, t \tag{8.36}$$

This implies that $\hat{g}_t^{(1)}(\hat{W}_t + B) \geq g_t^{(1)}(\hat{W}_t + B)$. We know that for $W = \hat{W}_t + B$,

$$\hat{g}_t^{(1)}(W) = tr\left\{W\Gamma_t W^T - 2W\Theta_t + C_t\right\} \tag{8.37}$$

where $\Gamma_t = t^{-1}\sum_{j=1}^{t} y_j y_j^T$, $\Theta_t = t^{-1}\sum_{j=1}^{t} y_j \hat{x}_j^T$, $C_t = t^{-1}\sum_{j=1}^{t} \hat{x}_j \hat{x}_j^T$, and $tr(\cdot)$ denotes the matrix trace. Similarly,

$$g_t^{(1)}(W) = tr\left\{W\Gamma_t W^T - 2W\hat{\Theta}_t + \hat{C}_t\right\} \tag{8.38}$$

where the matrix $\hat{\Theta}_t = t^{-1}\sum_{j=1}^{t} y_j\,(H_s(Wy_j))^T$, and $\hat{C}_t = t^{-1}\sum_{j=1}^{t} H_s(Wy_j)\,(H_s(Wy_j))^T$, with $W = \hat{W}_t + B$. Since $\hat{W}_t$ lives in a bounded set $\forall\ t$, and assuming a fixed B, we have that Γ_t, Θ_t, C_t, $\hat{\Theta}_t$, and $\hat{C}_t$ are all bounded. Therefore, we can find a convergent subsequence of $\left\{\hat{W}_t, \Gamma_t, \Theta_t, C_t, \hat{\Theta}_t, \hat{C}_t\right\}$. Let the subsequence be indexed by l_t $(t \geq 1)$, and let the accumulation point be $\left\{\hat{W}_\infty, \Gamma_\infty, \Theta_\infty, C_\infty, \hat{\Theta}_\infty, \hat{C}_\infty\right\}$. Taking the limit $t \to \infty$ on both sides of the inequality $\hat{g}^{(1)}_{l_t}(\hat{W}_{l_t} + B) \geq g^{(1)}_{l_t}(\hat{W}_{l_t} + B)$ immediately yields

$$\hat{g}^{(1)}_\infty(\hat{W}_\infty + B) \geq g^{(1)}_\infty(\hat{W}_\infty + B) = g^{(1)}(\hat{W}_\infty + B) \tag{8.39}$$

where the subscript ∞ denotes the corresponding accumulation point (of subsequence), and $g^{(1)}_\infty(\hat{W}_\infty + B) = g^{(1)}(\hat{W}_\infty + B)$ almost surely. The accumulation point $\hat{W}_\infty$ has full rank – otherwise the boundedness of the objective sequence $\left\{\hat{g}_{l_t}(\hat{W}_{l_t})\right\}$ would be violated.

To prove that $\hat{W}_\infty$ is a stationary point of $g(W)$, consider first order expansions of the functions in (8.39). We replace B by αB in (8.39) for small $\alpha > 0$. The function $\hat{g}^{(1)}_\infty(W)$ is (convex) quadratic with Lipschitz gradient and has a quadratic upper bound [10]. We also use a first order Taylor series expansion for $g^{(1)}(\hat{W}_\infty + \alpha B)$ (the existence of $\nabla g^{(1)}(\hat{W}_\infty)$ is proved in Appendix E.4) in (8.39) to get

$$\begin{aligned}\hat{g}^{(1)}_\infty(\hat{W}_\infty) + tr\left\{\alpha B^T \nabla \hat{g}^{(1)}_\infty(\hat{W}_\infty)\right\} + \frac{L}{2}\left\|\alpha B\right\|^2 \geq \\ g^{(1)}(\hat{W}_\infty) + tr\left\{\alpha B^T \nabla g^{(1)}(\hat{W}_\infty)\right\} + \epsilon\end{aligned} \tag{8.40}$$

where ϵ is the Taylor series remainder for the right hand side, and L is the Lipschitz constant mentioned in footnote 10 below.

We now show that $\hat{g}^{(1)}_\infty(\hat{W}_\infty) = g^{(1)}(\hat{W}_\infty)$ almost surely in (8.40). First, we use the fact that $\hat{g}^{(1)}_{l_t}(\hat{W}_{l_t})$ converges (by directly using (8.37)) to $\hat{g}^{(1)}_\infty(\hat{W}_\infty)$. Furthermore, similar to the function $u^{(1)}(y, W)$ in Lemma 38 of Appendix E.1, $g^{(1)}(W)$ is also Lipschitz on a compact set. Since $\hat{W}_{l_t}$ converges to $\hat{W}_\infty$, we therefore have that $g^{(1)}(\hat{W}_{l_t})$ converges to $g^{(1)}(\hat{W}_\infty)$. Therefore, using Proposition 11 (note that $\hat{g}^{(2)}_{l_t}(\hat{W}_{l_t})$ and $g^{(2)}(\hat{W}_{l_t})$ both converge to $\lambda_0\, v(\hat{W}_\infty)$,

[10] It satisfies $\hat{g}^{(1)}_\infty(W_2) \leq \hat{g}^{(1)}_\infty(W_1) + tr\left\{(W_2 - W_1)^T \nabla \hat{g}^{(1)}_\infty(W_1)\right\} + \frac{L}{2}\left\|W_2 - W_1\right\|_F^2\ \forall\ W_1, W_2$, and Lipschitz constant L.

where $\hat{W}_\infty$ has full rank), it is clear that $\hat{g}^{(1)}_\infty(\hat{W}_\infty) = g^{(1)}(\hat{W}_\infty)$ almost surely (in (8.40)).

Based on the above arguments, it is also clear that $g(\hat{W}_{l_t}) = g^{(1)}(\hat{W}_{l_t}) + g^{(2)}(\hat{W}_{l_t})$ converges (as $t \to \infty$) to $g(\hat{W}_\infty)$. Combining this with the result of Proposition 11, it is clear that $g(\hat{W}_\infty) = g^*$ with probability 1, where g^* is the (unique) limit of $g(\hat{W}_t)$ defined in Proposition 11. Thus, the accumulation point $\hat{W}_\infty$ achieves the value g^* of the expected cost with probability 1.

Now, upon substituting $\hat{g}^{(1)}_\infty(\hat{W}_\infty) = g^{(1)}(\hat{W}_\infty)$ into (8.40), then dividing (8.40) by $\left\|\alpha B\right\|$ ($B \neq 0$), and letting $\alpha \to 0$, we get that

$$tr\left\{B^T \nabla \hat{g}^{(1)}_\infty(\hat{W}_\infty)\right\} \geq tr\left\{B^T \nabla g^{(1)}(\hat{W}_\infty)\right\} \tag{8.41}$$

where we used the property of the Taylor series remainder that $\epsilon = o(\|\alpha B\|)$, so that $\lim_{\alpha\to 0}(\epsilon / \|\alpha B\|) = 0$. Since (8.41) holds (with probability 1) for any B, we must have $\nabla g^{(1)}(\hat{W}_\infty) = \nabla \hat{g}^{(1)}_\infty(\hat{W}_\infty)$ with probability 1. This also implies that $\nabla g(\hat{W}_\infty) = \nabla \hat{g}_\infty(\hat{W}_\infty)$.

Now, since $\hat{W}_{l_t}$ is a global minimizer of $\hat{g}_{l_t}(W)$ (for every t), we have that $\nabla \hat{g}_{l_t}(\hat{W}_{l_t}) = 0$ for each t. Moreover, $\nabla \hat{g}_{l_t}(\hat{W}_{l_t})$ converges to $\nabla \hat{g}_\infty(\hat{W}_\infty)$ as $t \to \infty$. Therefore, $\nabla \hat{g}_\infty(\hat{W}_\infty) = 0$. Since $\nabla \hat{g}_\infty(\hat{W}_\infty) = \nabla g(\hat{W}_\infty)$ with probability 1, we therefore have that $\nabla \hat{g}_\infty(\hat{W}_\infty) = \nabla g(\hat{W}_\infty) = 0$ with probability 1, or in other words, $\hat{W}_\infty$ is a stationary point of the function $g(W)$ with probability 1.

Since we worked with an arbitrary convergent subsequence in the above derivation, the result indicates that every accumulation point of $\left\{\hat{W}_t\right\}$ is a stationary points of g with probability 1, i.e., it satisfies first-order optimality conditions. Moreover, by the aforementioned arguments, every accumulation point achieves the same value (i.e., g^*) of the expected cost with probability 1. $\blacksquare$

The following corollary is based on Proposition 12.

Corollary 8. *Let Assumptions A1-A4 hold. Then, the distance between $\hat{W}_t$ and the set of stationary points of the expected cost $g(W)$ converges to 0 almost surely as $t \to \infty$.*

Proof: From Proposition 12, we know that every accumulation point of $\left\{\hat{W}_t\right\}$ is a stationary point of the expected cost $g(W)$ with probability 1.

Now, consider a convergent subsequence $\left\{\hat{W}_{l_t}\right\}$ of $\left\{\hat{W}_t\right\}$, that converges to a (nonsingular) limit $\hat{W}_\infty$. Then, $\nabla g(\hat{W}_\infty) = 0$ with probability 1.

Now, based on Lemma 41 of Appendix E.4, $\nabla g^{(1)}(\cdot)$ exists and is continuous at $\hat{W}_\infty$. Moreover, $\nabla g^{(2)}(\cdot)$ is also easily continuous at the nonsingular matrix $\hat{W}_\infty$. Combining these two results, it follows that $\nabla g(\cdot)$ is continuous at $\hat{W}_\infty$. Since $\left\{\hat{W}_{l_t}\right\}$ converges to $\hat{W}_\infty$, therefore, due to continuity $\left\{\nabla g(\hat{W}_{l_t})\right\}$ converges to $\nabla g(\hat{W}_\infty)$, which is 0 with probability 1.

The above results in fact imply that for any convergent subsequence of $\left\{\nabla g(\hat{W}_t)\right\}$, the limit is 0 with probability 1. Since 0 is the only limit (i.e., with probability 1) for any convergent subsequence, this implies that $\left\{\nabla g(\hat{W}_t)\right\}$ itself converges to 0 almost surely. From this, we can conclude that the distance between $\hat{W}_t$ and the set of stationary points of the expected cost $g(W)$ converges to 0 almost surely as $t \to \infty$. ■

This completes the proof.

8.5 Conclusions

In this chapter, we analyzed the convergence behavior of the newly proposed online sparsifying transform learning algorithms. We showed that our online transform learning algorithms are guaranteed to converge (almost surely) to the set of stationary points of the learning problem. Specifically, every accumulation point of the iterate sequence in our algorithm is a stationary point of the expected cost (the gradient of the cost $g(W)$ is zero at each point) with probability 1. Moreover, every accumulation point corresponds to the same expected cost ($g(W)$) value with probability 1. Our guarantee relies on few (easy to verify, handle) assumptions. Moreover, the result is for a highly non-convex problem, that is not simply biconvex. Further investigation of the local/global optimality of our scheme will be considered in future work.

CHAPTER 9

FAST DOUBLY SPARSE TRANSFORM LEARNING WITH CONVERGENCE GUARANTEES

9.1 Introduction

In this chapter [1], we focus on the learning of *square* sparsifying transforms $W \in \mathbb{R}^{n \times n}$, and develop a convex learning formulation. We also study non-convex variants of this convex formulation.

In Chapter 3, we investigated the learning of doubly sparse transforms [88, 2] W that are a product of two different transforms, i.e., $W = B\Phi$, where $\Phi \in \mathbb{R}^{n \times n}$ is an analytical transform with an efficient implementation, and $B \in \mathbb{R}^{n \times n}$ is a matrix that is constrained to be sparse (i.e., has few significant non-zero elements). The structure of $W = B\Phi$ was motivated by the fact that Φ matrices such as the DCT when applied to natural signals produce a result that is already approximately sparse. Thus, by further modifying the result using only a sparse transform B, one can produce a highly sparse result. Doubly sparse transforms can be learned, stored, and implemented efficiently. However, the doubly sparse transform learning formulations discussed in Chapter 3 were highly nonconvex, and the algorithms discussed therein for solving those problems lacked convergence guarantees (e.g., for the iterates).

9.1.1 Contributions

In this chapter, by exploiting a different set of properties (in particular, a skew-symmetry property is used to control transform conditioning) than in Chapter 3, we propose three formulations for doubly sparse transform learning, which are referred to as 1) 'convex' ($\ell_1 - \ell_1$) formulation, 2) 'partially

[1]Some parts of the material in this chapter appear in [164].

nonconvex' ($\ell_1 - \ell_0$) formulation, and 3) 'fully nonconvex' ($\ell_0 - \ell_0$) formulation.

Firstly, we propose a novel convex formulation for square doubly sparse transform learning. The doubly sparse structure enables fast implementation of the transform. The proposed convex formulation employs ℓ_1 penalties for sparsity of the transform (i.e., the B matrix in $W = B\Phi$) and sparse codes (of training data). This formulation has similarities to the well-known problem of compressed sensing (CS) [30, 31, 29] (see also [112, 113, 114, 115, 116, 117, 118, 119] for the earliest versions of CS). The proposed convex learning algorithm in this work is a scale-invariant version of standard FISTA [11], and is importantly guaranteed to converge to the global minimum of the cost, and moreover, converges quickly.

Second, we propose a 'partially non-convex' variant of the convex formulation by replacing (or, tightening) the ℓ_1 sparsity penalty on the sparse code with an ℓ_0 constraint. The motivation for such a tightening of the norm is to allow for exact control of sparsity (of the sparse code), which the ℓ_0 "norm" achieves. The proposed algorithm for solving the 'partially non-convex' formulation is guaranteed to converge to a good local minimum of the cost. Numerical experiments illustrate that the algorithm is also typically insensitive to initialization. In contrast to these convergence results for the convex or 'partially non-convex' schemes, the methods previously proposed in Chapter 3 lack such convergence guarantees.

Third, for comparison, we propose yet another formulation – a 'fully non-convex' version of our doubly sparse convex formulation, obtained by replacing the ℓ_1 penalties on both the sparse code and transform with ℓ_0 constraints.

Our numerical experiments indicate that all the proposed formulations give rise to significantly sparse and well-conditioned transforms in practice. The learned transforms provide better image representations than analytical transforms. Importantly, the convex formulation is observed to perform only slightly worse than the proposed non-convex variants here (i.e., the 'partially non-convex', or 'fully non-convex' formulations of this chapter). The 'partially non-convex' scheme here performs almost identically to the 'fully non-convex' one. However, the latter lacks a convergence guarantee. Lastly, the learned transforms obtained via the formulations proposed in this chapter perform comparably or somewhat worse (in image representation) than those learned using the non-convex (non-guaranteed) schemes of Chapter 3 in our experiments. This may be because we exploit a different set of (possibly more restrictive) properties (e.g., skew-symmetry) here than in Chapter 3. Nevertheless, we would like to emphasize that the schemes proposed in this chapter such as the convex or ‘partially non-convex’ schemes have the advantage of superior convergence guarantees.

9.1.2 Organization

The rest of this chapter is organized as follows. Section 9.2 introduces the (three) proposed problem formulations (referred to as the convex, ‘partially nonconvex’, and ‘fully nonconvex’ formulations) and their properties. Section 9.3 presents our transform learning algorithms. Section 9.4 describes the important convergence properties of the proposed algorithms as well as their computational cost. In Section 9.5, we present numerical experiments involving natural images that demonstrate the usefulness of our algorithms for image representation. In Section 9.6, we conclude with proposals for future work.

9.2 Problem Formulations and Properties

9.2.1 Prior Doubly Sparse Transform Learning

Given the training matrix $Y \in \mathbb{R}^{n\times N}$, we proposed to learn a square doubly sparse transform $W = B\Phi$ for the case of orthonormal Φ in Chapter 3 as follows:

$$\text{(P9.0)}\quad \min_{B,X} \; \|BZ - X\|_F^2 - \lambda \log\,|\det B| + \lambda \,\|B\|_F^2$$
$$s.t.\quad \|B\|_0 \leq r,\; \|X_i\|_0 \leq s \;\forall\, i$$

Here, $Z = \Phi Y$, and the columns X_i of the matrix $X \in \mathbb{R}^{n\times N}$ denote the sparse codes of the training signals in Y, with maximum allowed sparsity s. The term $\|BZ - X\|_F^2$ in (P9.0) denotes the sparsification error for the data Y in the transform $B\Phi$. The $-\log\,|\det B|$ and $\|B\|_F^2$ terms control the scaling and conditioning of B. The term $\|B\|_0 \triangleq \sum_{i,j} 1_{\{B_{ij}\neq 0\}}$, with B_{ij} the entry of B from row i and column j, and $1_{\{B_{ij}\neq 0\}}$ is the indicator function of $B_{ij} \neq 0$ (the indicator function takes value $+1$ when $B_{ij} \neq 0$, and value 0 otherwise).

Problem (P9.0) is, however, non-convex due to the log-determinant penalty and the ℓ_0 "norm" constraints. Experimental results for (P9.0) in Chapter 3 indicated that the learned B for natural images has an interesting structure of a positive diagonal and an approximately skew-symmetric off-diagonal. This structure is observed with various analytical Φ such as the DCT, Hadamard, Karhunen-Loève Transform (KLT), etc. The observed structure is a motivation for us to consider convex formulations for learning a doubly sparse transform.

9.2.2 Convex Learning Formulation

One could constrain B in (P9.0) to be positive definite, i.e., $B \succ 0$. If in addition the ℓ_0 "norm" constraints for B and X were relaxed to ℓ_1 norm constraints, or ℓ_1 penalties, the resulting problem would be jointly convex in B and X. However, upon further investigation, we found that enforcing B to be positive definite is too restrictive, and provides almost no improvement over the analytical Φ. Note that the results in [2] already indicate that the learned $B \not\succ 0$ for (P9.0), since it has an approximately skew-symmetric off-diagonal.

Here, we propose to model a sparse transform $B \in \mathbb{R}^{n\times n}$ as $I + A$, where I is the $n \times n$ identity, and A is a skew-symmetric matrix satisfying $A^T = -A$ (skew-symmetry implies that A has a zero diagonal), where $(\cdot)^T$ denotes the matrix transpose operation. This B corresponds to a transform $W = B\Phi = \Phi + A\Phi$, which is just the sum of the analytical Φ matrix and a deviation term $A\Phi$. For the proposed B, we have

$$B^T B = (I + A)^T (I + A) = (I - A)(I + A) = I - A^2 \tag{9.1}$$

Now, if A is small (has small Frobenius norm), then B is approximately orthonormal, since the second order term A^2 above can be considered negligible. Thus, the condition number of B can be controlled simply by controlling the magnitude of A.

Our convex problem formulation for doubly sparse transform learning is as follows:

$$\min_{A,X} \; \|(I+A)Z - X\|_F^2 + \frac{\eta}{4}\left\|A + A^T\right\|_F^2 + \mu\,\|A\|_1 + \xi\,\|X\|_1$$
$$s.t.\;\; A_{ii} = 0\,\forall\; i \qquad\qquad \text{(P9.1)}$$

Here, η, μ, and ξ are non-negative weights, and $Z = \Phi Y$ represents the transformed training data Y. The (trivial) condition $A_{ii} = 0\,\forall\, i$, although written as a constraint for simplicity, is in fact hard coded into the objective function, i.e., the optimization is only performed over the off-diagonal elements of A. Note that here $B = I + A$, where A has zeros on the diagonal, and is assumed to be approximately skew-symmetric. Approximate rather than exact skew-symmetry was observed in [2], and leads to slightly better performance in our experiments. Since A can be written as the sum of its orthogonal symmetric and skew-symmetric parts, i.e., $A = \frac{A+A^T}{2} + \frac{A-A^T}{2}$, the penalty $\frac{1}{4}\left\|A + A^T\right\|_F^2$ in (P9.1) helps ensure that the energy in the symmetric part is sufficiently small. The penalty $\|A\|_1 = \sum_{i,j}|A_{ij}|$ is to enforce sparsity of the off-diagonal of A. It also serves to keep A (magnitude) small, so that B is approximately orthonormal (i.e., is well-conditioned). Similarly, $\|X\|_1 = \sum_i \|X_i\|_1 = \sum_{i,j}|X_{ij}|$ ensures sparsity of the columns/elements of X. One could also alternatively replace the penalty $\xi\,\|X\|_1$ with $\sum_i \xi_i\,\|X_i\|_1$, when appropriate weights ξ_i are known. The penalty $\|(I+A)Z - X\|_F^2$ in (P9.1) measures the sparsification error of the data Y in the transform $(I+A)\Phi$.

While μ controls the magnitude of elements of A, with the optimal $\hat{A} \to 0$ in (P9.1) as $\mu \to \infty$, the parameters η and ξ can also have an interestingly similar effect. In particular, as $\eta, \xi \to \infty$ in (P9.1) (even with $\mu = 0$), we have that any optimal $\hat{A} = -\hat{A}^T$, and the optimal $\hat{X} = 0$. In this case, the sparsification error term is $||Z + \hat{A}Z||_F^2 = \|Z\|_F^2 + ||\hat{A}Z||_F^2$, which together with the $\mu||\hat{A}||_1$ term is minimal only at $\hat{A} = 0$ (assume Z has full row rank for this result to hold at $\mu = 0$).

(P9.1) is a linear least squares problem in X and the off-diagonal of A, with additional ℓ_1 norm regularizers, and is therefore, convex. It has a similar structure to the ℓ_1 norm-based compressed sensing problem [30, 31, 29]. We believe this is the first convex sparsifying transform learning formulation. However, the least squares part of the cost in (P9.1) is not strictly convex, since it has a linear variety of minimizers $(\hat{A}, \hat{X})$ satisfying $\hat{X} = (I + \hat{A})Z$, and $\hat{A} = -\hat{A}^T$. Thus, the ℓ_1 norm regularizers in (P9.1) can help ensure that the optimal minimizer(s) is sparse.

It is unclear whether the minimizer in (P9.1) is unique in general. One could modify Problem (P9.1) by replacing the ℓ_1 norms with smoothed ℓ_1 norms (for a matrix C, we define the smoothed ℓ_1 norm as $\sum_{i,j} \sqrt{C_{ij}^2 + \delta}$ with $\delta > 0$ sufficiently small), or by adding extra Frobenius norm penalties (for A, X) to the cost. This would ensure that the modified objective is always strictly convex (has unique minimizer). However, the modification requires introducing more parameters/weights. Hence, we do not pursue this approach here. Our experimental results suggest (see Section 9.5) that Problem (P9.1) may indeed have a unique global minimizer in practice.

Next, we discuss the interesting behavior of Problem (P9.1) with respect to the parameter ξ for fixed η and $\mu \neq 0$. For our analysis, we denote the overall objective function of (P9.1) by $F(A, X)$, i.e., $F(A, X) = \|Z + AZ - X\|_F^2 + \frac{\eta}{4}\left\|A + A^T\right\|_F^2 + \mu \|A\|_1 + \xi \|X\|_1$, where A has a zero diagonal. We also define a constant C_0 as follows.

$$C_0 = \min_{A:\, A_{ii}=0\, \forall\, i} F(A, 0) \tag{9.2}$$

For a matrix D, we define the soft-thresholding operator $S_{\xi/2}(\cdot)$ with threshold $\xi/2$ as

$$S_{\xi/2}(D) = \operatorname{sign}(D) \odot (|D| - \xi/2)_+ \tag{9.3}$$

where "$\odot$" represents element-wise multiplication between matrices, $\operatorname{sign}(\cdot)$ provides the signs of the elements of a matrix, and $(\cdot)_+$ zeros out all but the non-negative elements of a matrix. We now have the following result with proof.

Proposition 13. *Consider Problem (P9.1) with (transformed) data $Z = \Phi Y$ and fixed weights η and $\mu \neq 0$. Let σ_1 denote the largest singular value of Z, and let C_0 be defined as in equation* (9.2)*. Then, for $\xi \geq \xi_0 \triangleq 2\sigma_1\left(n + \frac{C_0}{\mu}\right)$, we have the optimal $\hat{X} = 0$ in (P9.1).*

Proof: Given Z, η and $\mu \neq 0$, let $(\hat{A}, \hat{X})$ denote a minimizer of (P9.1) with some fixed ξ. We then have

$$\mu\, ||\hat{A}||_1 \leq F(\hat{A}, \hat{X}) \leq F(A, 0) \tag{9.4}$$

where A is any matrix with a zero diagonal. The first inequality above follows from the non-negativity of the various terms in the objective of (P9.1). Equation (9.4) implies that

$$||\hat{A}||_1 \leq C_0/\mu \tag{9.5}$$

We also have the following inequalities.

$$|(Z+\hat{A}Z)_{ij}| \leq ||(I+\hat{A})Z||_F \leq \sigma_1 ||I+\hat{A}||_F \leq \frac{\xi_0}{2} \tag{9.6}$$

The last '$\leq$' inequality in (9.6) follows from $||I+\hat{A}||_F \leq ||I+\hat{A}||_1 = n + ||\hat{A}||_1$, and (9.5). The other inequalities in (9.6) are straightforward inequalities for norms. Since, the optimal $\hat{X} = S_{\xi/2}(Z+\hat{A}Z)$, equations (9.6) and (9.3) imply that $\hat{X} = 0$, when $\xi \geq \xi_0 = 2\sigma_1 \left(n + \frac{C_0}{\mu}\right)$. ■

Proposition 13 implies that given η and non-zero μ, the optimal $\hat{X}$ in (P9.1) becomes 0 at a finite $\xi = \xi_0$. It is also easy to see that when $\xi = 0$, the (unique) optimal $\hat{A} = 0$, $\hat{X} = (I+\hat{A})Z = Z$ in (P9.1), which corresponds to maximum sparsity of the sparse code (since there is no improvement over the original Z). Proposition 13 does not reveal the sparsity of the optimal $\hat{X}$ for $0 < \xi < \xi_0$. In practice, we observe that the sparsity drops to 0 from maximum value ($\|Z\|_0$) in a monotone manner (see Section 9.5).

9.2.3 Alternative Non-convex Formulations

We now propose non-convex variants of our convex formulation (P9.1), that allow for exact sparsity control. First, we propose a 'partially' non-convex variant of Problem (P9.1), where the ℓ_1 penalty on X is replaced by an ℓ_0 constraint.

$$\text{(P9.2)} \quad \min_{A,X} \; \|(I+A)Z - X\|_F^2 + \frac{\eta}{4}\left\|A + A^T\right\|_F^2 + \mu \, \|A\|_1$$
$$s.t. \;\; A_{ii} = 0 \; \forall \; i, \;\; \|X_j\|_0 \leq s \;\; \forall \; j$$

Similar to Problem (P9.0), each column of X is constrained to have a maximum sparsity of s in (P9.2). Thus, any optimal sparse code for (P9.2) always has at most s-sparse columns. Our problem (P9.2) has fewer degrees of non-convexity than (P9.0), since it lacks the log-determinant penalty and the ℓ_0

constraint on B.

For comparison, we also have the following 'fully' non-convex variant of Problem (P9.1), where the ℓ_1 penalties on both A and X are replaced by ℓ_0 constraints.

$$\begin{aligned} \text{(P9.3)} \quad & \min_{A,X} \; \|(I+A)Z - X\|_F^2 + \frac{\eta}{4} \left\|A + A^T\right\|_F^2 \\ & s.t. \;\; A_{ii} = 0 \, \forall \, i, \;\; \|A\|_0 \leq r', \;\; \|X_j\|_0 \leq s \;\; \forall \, j \end{aligned}$$

Here, the matrix A is allowed a maximum sparsity level of r'. Note that the ℓ_0 "norm" constraint on matrix A is invariant to scaling of A. However, the transform $B = I + A$ is non-trivially modified when A is scaled, which can adversely affect the sparsification error term in the cost of (P9.3). In practice, Problem (P9.3) gives rise to matrices A (see Section 9.5) whose elements have magnitude $\ll 1$, thereby making $B = I + A$ diagonally dominant (full rank) and also well-conditioned.

9.2.4 Relationship between Proposed Formulations

Problems (P9.1), (P9.2), and (P9.3) all have an analytical solution for X with fixed A [9]. We discuss that solution only for (P9.1) and (P9.2), because (P9.3) and (P9.2) are identical for fixed A. Now, in the case of (P9.1), the solution for X with fixed A is given as $X = S_{\xi/2}(Z + AZ)$ with $S_{\xi/2}(\cdot)$ representing the soft-thresholding operator defined in (9.3). In the case of Problem (P9.2), the solution for X with fixed A is obtained by zeroing out all but the s coefficients of largest magnitude in each column of $Z + AZ$ (when this solution is unique, it is equivalent to hard-thresholding $Z + AZ$ with a possibly different threshold for each column). The solutions for X with a fixed A in (P9.1) and (P9.2) are thus non-identical (except in extreme cases when they are both either 0 (for sufficiently large ξ and $s = 0$), or $Z + AZ$ (when $\xi = 0, s = n$)), since soft-thresholding always causes shrinkage of large coefficients, while hard-thresholding does not have such an effect. Thus, even if the optimal $\hat{A}$ for (P9.1) and (P9.2) are identical, the corresponding optimal sparse codes (for given $\hat{A}$) would be different in general. However, despite this non-equivalence, both (P9.1) and (P9.2) perform well and quite similarly in practice (see Section 9.5). The results of Section 9.5 indicate that the transforms A learned by the formulations (P9.1), (P9.2), and (P9.3), may themselves be quite similar. This is an interesting empirical observation that we plan to prove more rigorously in future work.

Our modeling of the sparse matrix B as $I + A$ with A approximately skew-symmetric results in learning formulations that are similar to compressed sensing formulations. Our problems are thus much easier to solve compared to the learning problems in the synthesis [17] and (noisy) analysis models [26]. For the latter models, the optimization problem with a fixed dictionary becomes a compressed sensing problem. The joint optimization over the dictionary and sparse code in those models is even harder.

9.3 Algorithms

We first discuss our algorithm for solving the convex Problem (P9.1), and then detail the algorithms for the non-convex (P9.2) and (P9.3).

9.3.1 DOSLIST - A Convex Transform Learning Algorithm

Our unconstrained convex minimization Problem (P9.1) has an objective that is the sum of smooth (ℓ_2) and non-smooth (ℓ_1) parts. Note that the smooth part is not strictly convex. Several algorithms have been proposed in the past for solving such a problem such as ISTA [165] (also see the references in [11]), FISTA [11], TWIST [166], NESTA [167], etc. The various algorithms (e.g., FISTA [11], NESTA [167]) can achieve a convergence rate of at best $O(1/k^2)$ for Problem (P9.1), where k refers to the iteration number. We propose a slightly modified version of FISTA for solving Problem (P9.1) here.

The algorithm that we propose for Problem (P9.1) is a scale-invariant version of standard FISTA [11] that uses multiple Lipschitz constants. It is specifically invariant (proven in Section 9.4) to scaling of the (transformed) data Z. We first outline some underlying assumptions/properties for our Problem, and then describe our algorithm. Let us define the function $f(A, X) = \|(I + A)Z - X\|_F^2 + \frac{\eta}{4} \left\|A + A^T\right\|_F^2$, where A is assumed to be zero on the main diagonal. Thus, $f(A, X)$ denotes the smooth part of the objective in (P9.1). Then, for any set of matrices $A \in \mathbb{R}^{n\times n}, A' \in \mathbb{R}^{n\times n}, X \in \mathbb{R}^{n\times N}, X' \in \mathbb{R}^{n\times N}$ where A and A' have zero diagonals, we assume that the function f satisfies the following inequality for appropriate constants L_A and L_X [2].

$$\begin{aligned} f(A', X') \leq \langle \nabla_A f(A, X), A' - A \rangle + \frac{L_A}{2} \|A' - A\|_F^2 \\ + f(A, X) + \langle \nabla_X f(A, X), X' - X \rangle + \frac{L_X}{2} \|X' - X\|_F^2 \end{aligned} \tag{9.7}$$

Here, $\nabla_A f$ and $\nabla_X f$ denote the gradients of f with respect to A (only its off-diagonal, since its diagonal is fixed to zero) and X, respectively. Both $\nabla_A f$ and $\nabla_X f$ are arranged in matrix form. For ∇_A to be an $n \times n$ matrix, we simply introduce zeros for its diagonal. Thus, we have

$$\nabla_X f(A, X) = 2(X - Z - AZ) \tag{9.8}$$

$$\nabla_A f(A, X) = G \odot (2(I + A)ZZ^T - 2XZ^T + \eta A + \eta A^T) \tag{9.9}$$

where G is a matrix of all ones and a zero main diagonal. The operator $\langle \cdot, \cdot \rangle$ in (9.7) denotes the standard trace inner product between matrices, that induces the Frobenius norm. Equation (9.7) is obviously satisfied when $L_A = L_X = L$, where L is a 'global' Lipschitz constant [11] of ∇f. However, L_A and L_X need not coincide in general (e.g., (9.7) may hold with $L_A = L$ and $L_X \ll L_A$).

Our algorithm for solving Problem (P9.1) called the DOubly Sparse Learning by Iterative Soft Thresholding (DOSLIST) algorithm, is presented in Fig. 9.1. It is similar to FISTA, but uses the block constants L_A and L_X satisfying (9.7). Importantly, the algorithm involves simple soft-thresholding-based updates. Note that we use fixed constants L_A and L_X in Fig. 9.1 for simplicity [3]. A simple choice for the initial A^0 in Fig. 9.1 is the zero matrix, which corresponds to the analytical sparsifying transform $W = (I + 0)\Phi = \Phi$. The initial X^0 is simply the soft thresholded version of $Z + A^0 Z$ (i.e., optimal X in (P9.1) for given $A = A^0$).

[2]Condition (9.7) (similar to the single Lipschitz constant case [168, 169]) follows from the assumption that $\langle \nabla_A f(A', X') - \nabla_A f(A, X),\, A' - A \rangle + \langle \nabla_X f(A', X') - \nabla_X f(A, X),\, X' - X \rangle \leq L_A \|A' - A\|_F^2 + L_X \|X' - X\|_F^2$.

[3]One could also obtain a version of DOSLIST with backtracking, similar to backtracking FISTA [11]. Backtracking FISTA finds a common constant L^k satisfying (9.7) at iteration k, which essentially corresponds to $\max(L_A^k, L_X^k)$. One can then decrease this constant for X alone (and vice-versa for A alone), until (9.7) is violated. This can determine $\min(L_A^k, L_X^k)$.

```
**Input :** $Z = \Phi Y$ - Data, $L_A$, $L_X$ - Constants satisfying equation (9.7), $J$ - number of iterations.
**Initialization :** $Q^1 = A^0$, $R^1 = S_{\xi/2}(Z + A^0 Z) = X^0$, $t_1 = 1$.
**For  k = 1:J Repeat**
    $A^k = S_{\mu/L_A}\left(Q^k - \frac{1}{L_A}\nabla_A f(Q^k, R^k)\right)$    (9.10)
    $X^k = S_{\xi/L_X}\left(R^k - \frac{1}{L_X}\nabla_X f(Q^k, R^k)\right)$    (9.11)
    $t_{k+1} = \frac{1 + \sqrt{1 + 4t_k^2}}{2}$    (9.12)
    $Q^{k+1} = A^k + \left(\frac{t_k - 1}{t_{k+1}}\right)\left(A^k - A^{k-1}\right)$    (9.13)
    $R^{k+1} = X^k + \left(\frac{t_k - 1}{t_{k+1}}\right)\left(X^k - X^{k-1}\right)$    (9.14)
**End**
```

Figure 9.1: DOubly Sparse Learning by Iterative Soft Thresholding (DOSLIST) Algorithm for Problem (P9.1). Note that the update of A^k above always leaves its diagonal to be 0.

9.3.2 DOSLAM Algorithm

For the (partially) non-convex Problem (P9.2), we propose an alternating algorithm similar to the one previously proposed for Problem (P9.0) [2]. We call our proposed algorithm as DOubly Sparse Learning by Alternating Minimization (DOSLAM).

In one step of the DOSLAM algorithm called the *Sparse coding step*, we solve (P9.2) with fixed A.

$$\min_X \ \|(I + A)Z - X\|_F^2 \ \ s.t. \ \ \|X_j\|_0 \leq s \ \forall \ j \tag{9.15}$$

The solution X, as discussed earlier (Section 9.3), is obtained exactly by zeroing out all but the s coefficients of largest magnitude in each column of $Z + AZ$. Thus, we write $X = \hat{H}_s(Z + AZ)$, where the operator $\hat{H}_s(\cdot)$ zeros out all but the s coefficients of largest magnitude (in case of non-uniqueness, choose these s coefficients as the ones with the lowest indices) in each column of a given matrix.

In the other step of the DOSLAM algorithm called the *Transform update* *step*, we solve (P9.2) with fixed X.

$$\min_{A; A_{ii}=0\, \forall i} \|(I+A)Z - X\|_F^2 + \frac{\eta}{4}\left\|A + A^T\right\|_F^2 + \mu \|A\|_1 \tag{9.16}$$

Here, as before (for (P9.1)), the condition $A_{ii} = 0$ is directly incorporated into the objective, making the above problem unconstrained. The objective function above is convex [4], and is similar to the objective of (P9.1) in that it involves linear least squares (smooth) penalties along with an ℓ_1 regularizer. Thus, it can be minimized using an iterative algorithm like ISTA [165, 11], or MFISTA [170] (which is a monotone version of FISTA). The use of either ISTA or MFISTA guarantees a monotone decrease in the objective function over iterations. This implies that the objective converges for the overall alternating algorithm DOSLAM, when employing either ISTA or MFISTA for transform update (see arguments in Lemma 46 of Appendix F.3). We will employ ISTA to obtain stronger convergence results in Section 9.4 and Appendix F.3. ISTA is guaranteed to converge to a global minimizer of the objective in (9.16) at $O(1/k)$ rate [11] [5]. Note that ISTA requires a Lipschitz constant as input. Since X is fixed in the transform update step, we are only interested in the Lipschitz constant of $\nabla_A f(A, X)$ (9.9) (with fixed X).

As we iterate over the sparse coding and transform update steps for (P9.2), we initialize the ISTA with the previously (or, most recently) updated A. DOSLAM also requires an initial A^0 that has an all zero main diagonal (e.g., 0). The DOSLAM algorithm is outlined in Figure 9.2.

9.3.3 DOSLIHT Algorithm

For the (fully) non-convex Problem (P9.3), we propose an alternating algorithm called DOubly Sparse Learning by Iterative Hard Thresholding (DOSLIHT). Our algorithm alternates between solving for X and A.

In the *Sparse coding step* of the DOSLIHT algorithm, we solve (P9.3) with

[4]When the matrix $Z = \Phi Y$ additionally has full row rank, then the term $\|(I+A)Z - X\|_F^2$ is strongly convex with respect to the off-diagonal elements of A. In this special case, the objective of (9.16) is also strictly/strongly convex.

[5]MFISTA has a better $O(1/k^2)$ convergence rate. However, the iterations of ISTA are computationally faster than those of MFISTA. Hence, even though ISTA typically requires more iterations to reach a similar accuracy as MFISTA, the overall DOSLAM algorithm was observed to be slightly faster in our experiments when employing ISTA rather than MFISTA (while achieving the same image representation accuracy).

```
Input : $Z = \Phi Y$ - Data, $L$ - A Lipschitz Constant of $\nabla_A f(A, X)$ in (9.9), $J$ - number of alternations, $M$ - number of ISTA iterations for transform update.
Initial Estimates : $A^0$ - initial $A$, $X^0$ - initial $X$ (e.g., $\hat{H}_s(Z + A^0 Z)$).
For k = 1:J Repeat
    Initialization for ISTA : $Q^0 = A^{k-1}$.
    For i = 1:M Repeat
        $Q^i = S_{\mu/L}\left(Q^{i-1} - \frac{1}{L}\nabla_A f(Q^{i-1}, X^{k-1})\right)$    (9.17)
    End
    $A^k = Q^M$    (9.18)
    $X^k = \hat{H}_s(Z + A^k Z)$    (9.19)
End
```

Figure 9.2: DOubly Sparse Learning by Alternating Minimization (DOSLAM) Algorithm for Problem (P9.2). Since the initial A^0 has a zero main diagonal, the update of A^k above always leaves its diagonal to be 0.

fixed A. Since (P9.3) and (P9.2) are identical for a fixed A, the sparse coding solution X is exactly given as $X = \hat{H}_s(Z + AZ)$, where the operator $\hat{H}_s(\cdot)$ was defined for the DOSLAM algorithm.

In the *Transform update step* of the DOSLIHT algorithm, we solve for (P9.3) with fixed X. The resulting optimization problem is as follows.

$$\begin{aligned} \min_{A;A_{ii}=0\,\forall i} \quad & \|(I+A)Z - X\|_F^2 + \frac{\eta}{4}\left\|A + A^T\right\|_F^2 \\ s.t. \quad & \|A\|_0 \le r' \end{aligned} \tag{9.20}$$

Here, as before (for (P9.1)), the optimization is only performed with respect to the off-diagonal of A (the diagonal is fixed). The above problem has a linear least squares objective with an ℓ_0 sparsity constraint. We solve this problem using the well-known IHT algorithm [102]. The iterative updates using IHT are as follows.

$$A^{i+1} = H_{r'}\left(A^i - \alpha_A \nabla_A f(A^i, X)\right) \tag{9.21}$$

Here, i denotes the iteration number, and α_A is a step-size. The operator $H_{r'}(\cdot)$ zeros out all but the r' elements of largest magnitude (in case of

```
Input : $Z = \Phi Y$ - Data, $\alpha_A$ - step size used within IHT, $J'$ - number of alternations, $M'$ - number of IHT iterations in transform update.
Initial Estimates : $A^0$ - initial $A$, $X^0$ - initial $X$ (e.g., $\hat{H}_s(Z + A^0 Z)$).
For k = 1:$J'$ Repeat
    Initialization for IHT : $Q^0 = A^{k-1}$.
    For i = 1:$M'$ Repeat
        $Q^i = H_{r'}\left(Q^{i-1} - \alpha_A \nabla_A f(Q^{i-1}, X^{k-1})\right)$    (9.22)
    End
    $A^k = Q^{M'}$    (9.23)
    $X^k = \hat{H}_s(Z + A^k Z)$    (9.24)
End
```

Figure 9.3: DOubly Sparse Learning by Iterative Hard Thresholding (DOSLIHT) Algorithm for Problem (P9.3). Since the initial A^0 has a zero main diagonal, the update of A^k above always leaves its diagonal to be 0.

non-uniqueness, choose the ones with the lowest indices) in a given matrix. The function f and gradient $\nabla_A f$ used above were already defined for the DOSLIST algorithm. Although we base our transform update procedure on IHT (for its computational simplicity), one could also alternatively use other algorithms such as HTP [171], etc.

The DOSLIHT algorithm requires an initial A^0 (e.g., 0) and X^0. The overall algorithm is outlined in Figure 9.3.

9.4 Algorithm Properties and Computational Cost

We first describe the convergence and scale-invariant behavior of the DOSLIST algorithm that solves Problem (P9.1), and then detail the convergence properties of the non-convex learning schemes (DOSLAM, DOSLIHT). In the end, we briefly outline the computational costs of the algorithms.

9.4.1 Convergence of DOSLIST

The following theorem provides the $O(1/k^2)$ convergence rate of the DOSLIST algorithm, that minimizes the convex objective $F(A, X)$ of Problem (P9.1).

Theorem 6. *Let $\{A^k\}$, $\{X^k\}$, $\{Q^k\}$, $\{R^k\}$ denote the iterate sequences generated by the DOSLIST algorithm for data Z. Further, let (A^*, X^*) denote any minimizer of Problem (P9.1). Then, for $\forall k \geq 1$, we have*

$$F(A^k, X^k) - F(A^*, X^*) \leq \frac{2C}{(k+1)^2} \tag{9.25}$$

where the constant $C = L_A \|A^0 - A^*\|_F^2 + L_X \|X^0 - X^*\|_F^2$.

The proof of the above theorem is similar to that of FISTA [11], except that we use the block-Lipschitz property as defined by equation (9.7). The details of the proof are presented in Appendix F.1.

The constant C in (9.25) can also be upper bounded as follows. Since, $X^0 = S_{\xi/2}(Z + A^0 Z)$ and $X^* = S_{\xi/2}(Z + A^* Z)$, we have

$$\left\|X^0 - X^*\right\|_F^2 \leq \left\|A^0 Z - A^* Z\right\|_F^2 \leq \sigma_1^2 \left\|A^0 - A^*\right\|_F^2$$

where the first inequality above follows from the non-expansiveness property of the soft-thresholding operator. Thus, a bound on C that is free of the X variable is

$$C \leq (L_A + L_X \sigma_1^2) \left\|A^0 - A^*\right\|_F^2 \tag{9.26}$$

9.4.2 Scale-invariance of DOSLIST

We now show the interesting scale-invariance behavior of the DOSLIST algorithm. We first show how the Problem (P9.1), and the constants L_A and L_X in (9.7) modify, when the data Z is scaled. For our analysis, we introduce Z into our previous notation (in Sections 9.2 and 9.3) and write $f(A, X, Z) = \|(I + A)Z - X\|_F^2 + \frac{\eta}{4}\left\|A + A^T\right\|_F^2$. Similarly, the overall objective of (P9.1) is $F(A, X, Z)$.

Let us call the objective of (P9.1) with data Z and weights η, μ, ξ, i.e., $F(A, X, Z)$, as the 'un-scaled' objective. When Z is scaled by a non-zero scalar $\alpha \in \mathbb{R}$ in (P9.1), we also need to scale the weights η and μ by α^2, and ξ by $|\alpha|$. Then, by replacing X with $\alpha X''$, the new objective $\widetilde{F}$ with αZ and scaled weights satisfies

$$\widetilde{F}(A, X, \alpha Z) = \alpha^2 F(A, X'', Z) \tag{9.27}$$

Therefore, both F and $\widetilde{F}$ have the same set of minimizers with respect to A. Moreover, the minimizers with respect to X for $\widetilde{F}$ are α times the corresponding minimizers for F (this makes perfect sense since the data Z was trivially scaled by α).

The following lemma shows the behavior of the constants L_A and L_X with scaling.

Lemma 5. *Suppose that L_A and L_X are the constants satisfying* (9.7) *for the function $f(A, X, Z)$ with data Z and weight η. For any non-zero scalar $\alpha \in \mathbb{R}$, if we replace Z with αZ, and η with $\alpha^2 \eta$, then the new Lipschitz constants $L_A^{'}$ and $L_X^{'}$ corresponding to the modified function $\widetilde{f}$ satisfy*

$$L_A^{'} = \alpha^2 L_A,\ L_X^{'} = L_X. \tag{9.28}$$

The proof of the above lemma is provided in Appendix F.2. We can also write the Lipschitz constants as $L_A = C_1 \sigma_1^2$, $L_X = C_2$, where σ_1 is the largest singular value of Z, and C_1, C_2 are the constants satisfying (9.7) when Z is scaled to have unit spectral norm. This form of L_A and L_X models their behavior with respect to scaling of Z, as dictated by Lemma 5. Although the constants C_1 and C_2 could perhaps vary for different (not related by a scaling) data-sets Z, we derive estimates (bounds) in Appendix F.4 that work for any Z.

We now use Lemma 5 to show that the DOSLIST algorithm is scale-invariant. Let $\{A^k\}$ and $\{X^k\}$ be the DOSLIST iterate sequences obtained with the (un-scaled) input Z, weights η, μ, ξ, and constants L_A, L_X satisfying (9.7). Further, let $\{A_1^k\}$ and $\{X_1^k\}$ denote the modified sequences generated by the DOSLIST algorithm with input αZ, and appropriately scaled weights (i.e., η, μ, and ξ scaled by α^2, α^2, and $|\alpha|$ respectively) and appropriately scaled Lipschitz (L_A scaled by α^2 and L_X left un-scaled) constants. Then, by looking at the effect of the scaling on each step in Figure 9.1, it is easy to observe that $A_1^k = A^k$ and $X_1^k = \alpha X^k$. Thus, the DOSLIST algorithm always generates the same sequence $\left\{A^k\right\}$ irrespective of the scaling on Z.

It is also interesting to observe the effect of scaling on the bound (9.25) in Theorem 6. When Z is scaled by α in (P9.1) (and the weights, Lipschitz constants are also scaled appropriately), the constant C in (9.25) simply gets scaled by α^2. This is because L_A gets scaled by α^2, and X^0 and X^* get scaled by α each, whereas A^0, A^* (by the result of equation (9.27)), and L_X don't scale. Moreover, using the above results and equation (9.27), the left hand side (LHS) of (9.25) corresponding to the data αZ and scaled weights [6] is also α^2 times the LHS corresponding to the un-scaled Z. Thus, scaling Z by α simply causes (9.25) to scale by α^2 throughout. If we divide (9.25) by the non-negative term $F(A^*, X^*)$ (assuming it is non-zero), then the resulting equation is in fact, scale-invariant.

We now show that the standard FISTA [11] is scale-dependent. Standard FISTA is equivalent to using a single $L = \max(L_A, L_X)$ (this is the smallest L for which equation (9.7) becomes equal to the corresponding condition for f in FISTA [11]) in DOSLIST. For example, consider the case when $\sigma_1(Z)$ (the scaling) is sufficiently large, then $L = \max(C_1\sigma_1^2, C_2) = C_1\sigma_1^2$, and it is easy to see that the steps of standard FISTA (Fig. 9.1 with $L_X = L_A = L$) are not scale-invariant (i.e., if a particular choice of σ_1 gets scaled by a factor (e.g., 2), it results in a completely new/unrelated iterate sequence). Moreover, it can be shown for this case that the bound in Theorem 6 (with single L) is also not homogeneous to scaling, and the constant C scales badly as $O(\sigma_1^4)$. In practice, we observed that standard FISTA (with either constant step size or backtracking) has a poor (slow) convergence behavior for (P9.1), unless the scaling of Z is manually tuned for better convergence.

9.4.3 Convergence of DOSLAM

Now, we detail the convergence behavior of our alternating algorithm DOSLAM, for Problem (P9.2). Problem (P9.2) has the constraint $\|X_j\|_0 \leq s \,\forall\, j$, which can instead (equivalently) be added as a penalty in the objective by using a barrier function $\psi(X)$ (which takes the value $+\infty$ when the constraint is violated, and is zero otherwise). In this form, Problem (P9.2) is

[6] With scaling $F(A^k, X^k, Z)$ in (9.25) gets modified as $\widetilde{F}(A_1^k, X_1^k, \alpha Z) = \widetilde{F}(A^k, \alpha X^k, \alpha Z)$ which is simply $\alpha^2 F(A^k, X^k, Z)$ by equation (9.27). The effect of scaling on the $F(A^*, X^*, Z)$ term in (9.25) is similar.

unconstrained (note no optimization over diagonal of A), and we denote its objective as $g(A, X)$. For a matrix U, we let $\beta_j(U_i)$ denote the magnitude of the j^{th} largest element (magnitude-wise) of the column U_i (i^{th} column of U). For some matrix C, $\|C\|_\infty \triangleq \max_{i,j} |C_{ij}|$. We then have the following Theorem (similar to the result in Chapter 4) on the convergence of the algorithm for (P9.2).

Theorem 7. *Let $\{A^k, X^k\}$ denote the iterate sequence generated by the DOSLAM algorithm for (P9.2) with data Z. Then, the objective sequence $\{g(A^k, X^k)\}$ is monotone decreasing, and converges to a finite value, say $g^* = g^*(A^0, X^0)$. Moreover, the iterate sequence is bounded, and every accumulation point (A, X) of the iterate sequence is a fixed point of the algorithm satisfying the following local optimality condition.*

$$g(A + dA, X + \Delta X) \geq g(A, X) = g^* \tag{9.29}$$

The condition holds for all $dA \in \mathbb{R}^{n\times n}$ with a zero diagonal, and all $\Delta X \in \mathbb{R}^{n\times N}$ satisfying at least one of the following conditions: 1) $tr\left\{(Z + AZ - X)\Delta X^T\right\} \leq 0$; 2) $\|\Delta X\|_\infty < \min_i \{\beta_s(U_i) : \|U_i\|_0 > s\}$, where $U = (I + A)Z$. Moreover, if $\|U_i\|_0 \leq s \,\forall\, i$, then ΔX can be arbitrary.

Theorem 7 indicates local convergence of our alternating DOSLAM algorithm. Every accumulation point (A, X) of the DOSLAM iterate sequence is a local optimum by equation (9.29), and satisfies $g(A, X) = g^* \triangleq g^*(A^0, X^0)$. Thus, for a particular initial (A^0, X^0), all the accumulation points of the iterates are equivalent (in terms of their cost), or equally good local minima [7]. We can say that the objective converges to a local minimum for DOSLAM.

The condition (9.29) holds for the algorithm irrespective of initialization. However, the local minimum g^* that the cost converges to may possibly depend on initialization. The condition (9.29) also holds irrespective of the number of iterations (M) of ISTA in the transform update step of DOSLAM. We empirically observed that the choice of M only affects the speed of convergence of DOSLAM (i.e., faster convergence for larger M). The local optimality (9.29) holds no matter how large the perturbation dA is. Moreover,

[7]Our experiments didn't indicate any large oscillations in the iterates (after many iterations), indicating that the accumulation point may be unique in practice.

the set of perturbations ΔX, for which (9.29) holds is also large (union of a half space and a local region). All the above properties indicate a somewhat strong convergence ('almost global') for our proposed DOSLAM algorithm. In practice (see Section 9.5), DOSLAM is insensitive to intialization indicating potential global convergence. The proof of Theorem 7 is presented in Appendix F.3.

Our convergence result for the non-convex learning algorithm DOSLAM, is free of any extra conditions/requirements (compare to algorithms such as IHT [102, 103] that solve non-convex problems but require extra stringent conditions for their convergence results to hold).

An interesting fact that follows trivially from Theorem 7 is that if an accumulation point (A, X) of the DOSLAM iterate sequence satisfies $\|U_i\|_0 \leq s \, \forall \, i$ with $U = (I + A)Z$, then that accumulation point is a global minimizer for Problem (P9.2). It then also follows that the objective converges to its global minimum in this case.

9.4.4 Convergence of DOSLIHT

The DOSLIHT algorithm alternates between sparse coding and transform update steps. The sparse coding step has an exact solution by thresholding. The transform update step (9.20) is solved using IHT. The IHT algorithm is guaranteed to converge to a global minimum [102] of (9.20) under certain strict conditions, and to a local minimum [103] under milder conditions.

For the overall alternating DOSLIHT algorithm, we conjecture that convergence results similar to DOSLAM may hold under some extra conditions (similar to [102, 103]). However, we do not pursue the full convergence analysis of DOSLIHT in this work. In practice, DOSLIHT is insensitive to initialization, and performs (see Section 9.5) similar to DOSLAM in image representation.

9.4.5 Computational Cost

The proposed algorithms for (P9.1), (P9.2), and (P9.3) have a low computational cost, which scales in order as $O(n^2 N)$ per-iteration. This cost can be easily derived similarly to the one for (P9.0) [2]. All the proposed algorithms involve operations with sparse matrices, which can be implemented very efficiently (similar to [2]).

9.5 Numerical Experiments

In this section, we first outline the framework for our experiments, and then provide the results.

9.5.1 Framework

We illustrate the convergence behavior of the proposed transform learning schemes in this section. We also study the effectiveness of the proposed learning schemes for representing images. For the latter task, we work with six different 512×512 images shown in Fig. 9.4. In all our experiments, we learn sparsifying transforms $B\Phi$ from the $\sqrt{n} \times \sqrt{n} = 12 \times 12$ (zero mean) non-overlapping patches of the images, with Φ being the patch-based 2D DCT [9]. The means (or DC values) of the patches are removed and we only sparsify the mean-subtracted patches which are stacked as columns of Y (patches represented as vectors). The means can be added back for image display. We learn the sparse transforms $B = I + A$ using the proposed algorithms for (P9.1), (P9.2), and (P9.3).

As part of the empirical demonstration of convergence behavior, we will show that the DOSLIST algorithm for (P9.1) converges to the same global solution irrespective of initialization. We will also plot the sparsity of the learned X in (P9.1) as a function of the parameter ξ to illustrate the behavior of sparsity with respect to that parameter (see our Proposition 13). Apart from illustrating the convergence of our algorithms for (P9.2) and (P9.3), we will also show that the algorithms are insensitive to initialization. We also illustrate the similarity of the transforms learned using the various proposed formulations.

For the task of image representation, we compare the performance of our learned transforms (via (P9.1), (P9.2), and (P9.3)) to those learned using the alternating algorithm [2] that solves (P9.0), and the fixed patch-based 2D DCT itself. We measure the quality of the transforms $W = B\Phi$ using the normalized sparsification error (NSE), and recovery peak signal to noise ratio (rPSNR) metrics (see Chapter 3). The NSE metric [9] is defined as $\|WY - X\|_F^2 \,/\, \|WY\|_F^2$, and it measures the fraction of energy lost in sparse fitting in the transform domain. For our comparisons, we use X obtained by thresholding each column of WY at sparsity s for all the algorithms. The rPSNR metric is defined (in decibels (dB)) as the scaled (by the factor 20) base-10 logarithm of $255\sqrt{P}/ \|Y - W^{-1}X'\|_F$, where P is the number of image pixels. While we can use X' obtained by just thresholding WY (at sparsity s per-column) [9], we found that the rPSNR improves by obtaining only the support of X' as the indices of the non-zeros of the thresholded WY, and then performing a simple least squares update of X' (column by column) on the known support [8] (similar to the Hard Thresholding Pursuit algorithm [171]). rPSNR is a simple surrogate for the compression performance of transforms.

For all our experiments, the various parameters for (P9.1), (P9.2), and (P9.3) are set [9] as $n = 144$, $\eta = (3.26 \times 10^{-3})\sigma_1^2(Z)$, $\mu = (2.18 \times 10^{-6})\sigma_1^2$, $s = 24$. We use a different adaptive $\xi_i = \xi_1 \times \beta_{s+1}(Z_i) = 0.44 \times \beta_{s+1}(Z_i)$ (scales linearly with $Z = \Phi Y$ assuming $\beta_{s+1}(Z_i) \neq 0 \,\forall\, i$) for each patch (column of X) in (P9.1). While we could use the upper bounds (estimates) for L_A, L_X, and L derived in Appendix F.4, we found that our algorithms converge slightly faster with slightly lower values for these constants, which we obtain empirically. We use constants $L_A = 2.56\sigma_1^2$ and $L_X = 3.7$ (for DOSLIST), and $L = 1.42\sigma_1^2$ (for DOSLAM), which we found empirically [10]. For (P9.3), we allow a transform sparsity of $\|I + A\|_0 \leq 0.25 \times n^2$, and the step size $\alpha_A = 1/2\sigma_1^2$ in DOSLIHT. For (P9.2) and (P9.3), we run 100 iterations of ISTA and IHT within the transform update steps, respectively. The initial $A^0 = 0$ for all our algorithms unless mentioned otherwise. The parameters for (P9.0) are set as $\lambda = 8.7 \times 10^{-3}\sigma_1^2$, and $r = 0.25 \times n^2$ (n, s are set the same as for (P9.2)). Other settings in the algorithm for (P9.0) are as in [2].

[8]The least squares update is easily obtained by solving $\min_{X_i'} \|Y_i - W^{-1}X_i'\|_2$ for each i, with the locations of the non-zeros in the X_i' fixed to those in the thresholded (at sparsity s) WY_i.

[9]The settings chosen here work well in our experiments. However, our algorithms' performance may be even better with some optimal tuning of the parameters.

[10]Note that one could alternatively use backtracking.

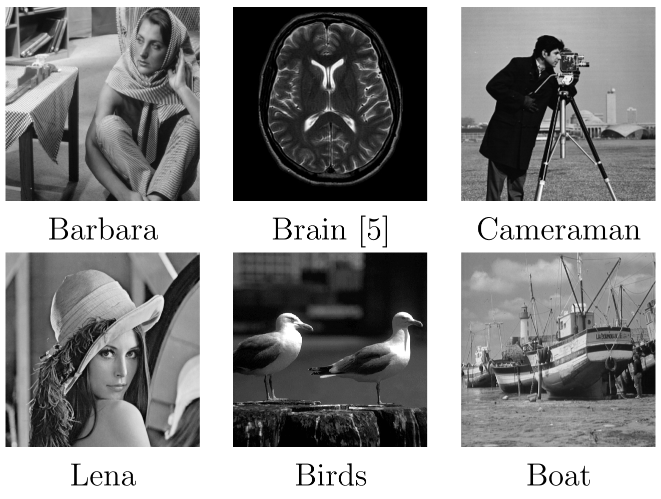


Figure 9.4: The test images.

9.5.2 Convergence Behavior of Proposed Algorithms

We first illustrate the convergence behavior of the DOSLIST algorithm for Problem (P9.1). We work with the 512×512 Cameraman image shown in Fig. 9.4. Figure 9.5(a) plots the objective of (P9.1) for different initializations. We consider 4 different initializations for the off-diagonal elements of A - the first one is the zero initialization, and the other three are random Gaussian initializations with zero mean and standard deviations of 0.001, 0.01, and 0.05, respectively. The objective is seen to converge to the same value for all initializations [11]. This is expected from the global convergence Theorem 6. The transforms $B = I + A$ learned using various initializations are shown in Figures 9.5(b)-(d) (magnitudes are shown), and they are seen to be sparse. For example, the transform $I + A$ learned using the zero initialization for A has an ℓ_0 sparsity of $0.53 \times n^2$, and moreover, it also has many more elements that are close to zero (only 20% of the elements have magnitude$>$ 0.003). Interestingly, the transforms learned with the various initializations turn out to be identical. This observation leads us to conjecture that Problem (P9.1) may have a unique global minimizer in practice. The learned transforms with different initializations are all well-conditioned (condition number of 1.89).

We plot in Figure 9.5(e), the evolution of $\left\|B^k - B^{k-1}\right\|_F / \left\|B^{k-1}\right\|_F$, where k denotes the algorithm iteration number and $B^k = I + A^k$. The plot is for the case of zero initialization for A. The plotted quantity measures the relative

[11]Worse initializations require more algorithm iterations.

change between successive iterates/transforms. It quickly decreases to a low value, and could be used as a stopping condition for the algorithm.

Figure 9.5(f) plots the percentage sparsity ($\|X\|_0 / nN$ in percent) of the sparse code X output from the DOSLIST algorithm, as a function of the parameter ξ_1 (defined in Section 9.5.1). At $\xi_1 = 0$, the observed sparsity is 100%. As ξ_1 is increased to a large value, the sparsity of the sparse code output X drops to an exact zero (e.g., at $\xi_1 \geq 1000$ in the plot). This isn't surprising, since Proposition 13 predicts that the sparsity of the optimal X in (P9.1) goes to zero at a finite ξ_0 [12]. However, in the region $0 < \xi_1 < 1000$, the sparsity of the optimal X in (P9.1) can now be seen to monotonically decrease to 0. Although we only plot the exact sparsity percentage here, many elements of the learned X tend to be not exactly zero, but still negligible in magnitude.

Finally, in Figure 9.5(g), we plot the objectives for DOSLIST and standard FISTA [11] (when standard FISTA is applied to our Problem (P9.1)), for the case of zero initialization for A. For standard FISTA, we employed backtracking search, with the parameter $\eta = 1.1$ [11] (note that this η is not our skew-symmetry weighting parameter) for the backtracking. Standard FISTA performs poorly compared to DOSLIST here, as is clear from Figure 9.5(g). This is because the performance of FISTA is scale-dependent as discussed in Section 9.4.

Next, we discuss the convergence behavior of the DOSLAM and DOSLIHT algorithms that solve the non-convex problems (P9.2) and (P9.3), respectively. Figure 9.6(a) plots the objective of (P9.2) for various initializations. Similar to (P9.1), we consider 4 different initializations for the off-diagonal elements of A - the first one being the zero initialization, and the other three being random Gaussian initializations with zero mean and standard deviations 0.001, 0.01, and 0.05, respectively. The objective of (P9.2) is seen to converge to the same value for the various initializations. Although not shown here, the objective of (P9.3) also has a similar behavior for the various initializations. This indicates that the DOSLAM and DOSLIHT algorithms are insensitive to initialization. For the DOSLAM algorithm, this observation together with Theorem 7 may suggest potential global convergence (in

[12]The result of Proposition 13 holds even when a different weight ξ_i is used for each column of X. The only difference is that the condition $\xi \geq \xi_0$ in the proposition gets replaced by $\xi_i \geq \xi_0 \forall\, i$.

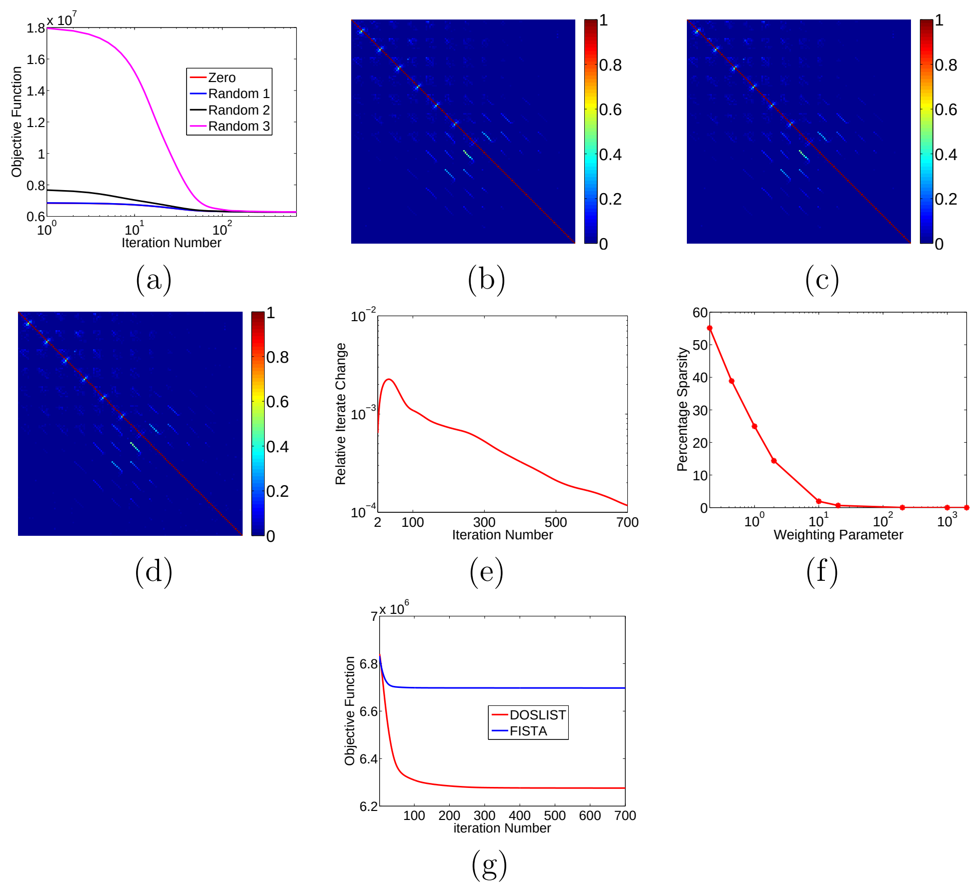


Figure 9.5: Behavior of the DOSLIST algorithm for (P9.1): (a) Objective function vs iterations for the cases of zero, Random 1 (standard deviation of 0.001), Random 2 (standard deviation of 0.01), and Random 3 (standard deviation of 0.05) initializations (for A), (b) Magnitude of learned $B = I + A$ with zero initialization, (c) Magnitude of learned $B = I + A$ with random Gaussian initialization (standard deviation of 0.01), (d) Magnitude of learned $B = I + A$ with random Gaussian initialization (standard deviation of 0.05), (e) Plot of relative iterate change for the case when zero initialization is used for A, (f) Plot of percentage sparsity of learned X (i.e., $\frac{\|X\|_0}{nN}$ in percent) as a function of the weighting parameter ξ_1, (g) Objective function for DOSLIST and standard FISTA [11] (with backtracking) for the case of zero initialization of A.

terms of the objective) in practical scenarios.

Figures 9.6(b) and 9.6(c) show the learned transforms (magnitude) $I + A$ obtained via (P9.2) and (P9.3), when employing the zero initialization for A. These transforms are very similar and have a small relative (Frobenius) error of 6%. They also appear somewhat similar to the transforms shown

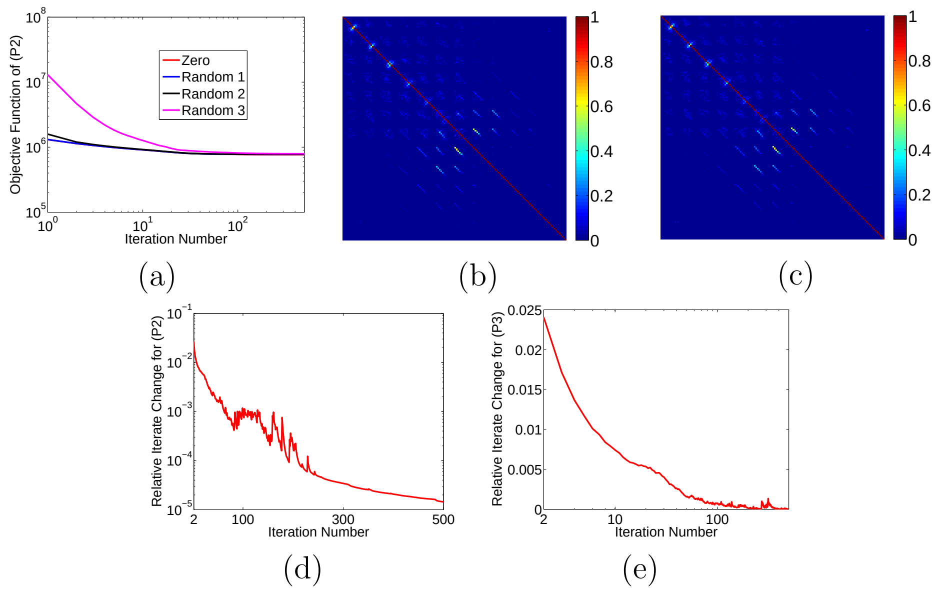


Figure 9.6: Behavior of the DOSLAM and DOSLIHT algorithms for (P9.2) and (P9.3): (a) Objective function vs iterations for (P9.2) for the cases of zero, Random 1 (standard deviation of 0.001), Random 2 (standard deviation of 0.01), and Random 3 (standard deviation of 0.05) initializations (for A), (b) Magnitude of learned $B = I + A$ for (P9.2) with zero initialization, (c) Magnitude of learned $B = I + A$ for (P9.3) with zero initialization, (d) Plot of relative iterate change for (P9.2) for the case when zero initialization is used for A, (e) Plot of relative iterate change for (P9.3) for the case when zero initialization is used for A.

for (P9.1) in Figure 9.5. The transform in Figure 9.6(b) for (P9.2) has an ℓ_0 sparsity of $0.52 \times n^2$, and moreover, has only 23% of its elements with magnitude above 0.003.

Figures 9.6(d) and 9.6(e) plot the relative iterate changes for (P9.2) and (P9.3), for the zero initialization case. The relative iterate change is seen to decrease to a low value in both cases. This behavior in the case of (P9.2) is predicted in Lemma 51 of Appendix F.3.

9.5.3 Comparison of Learning Schemes

Here, we compare the various transform learning schemes for representing the images in Figure 9.4. We learn adaptive transforms via (P9.0), (P9.1), (P9.2), and (P9.3) for each image. We stop the iterations of each of the algorithms when the relative iterate change falls below a low value of 0.01%.

We also additionally include a fixed maximum iteration count of 300 for all the algorithms. The learned transforms in all cases are well-conditioned with condition numbers between 1 to 4.

For (P9.1) and (P9.2), we generate exactly sparse transforms at a sparsity level of $0.25 \times n^2$ by thresholding the learned ones (learned B). Note that the transforms learned via (P9.1) and (P9.2) are already exactly sparse. However, they typically also contain many elements close to zero, which can be thresholded, without affecting the transform quality, but improving its sparsity.

Tables 9.1 and 9.2 provide the values of the performance metrics (i.e., NSE and rPSNR) for the various algorithms, along with the corresponding values for the 2D DCT Φ for the six images. The learned transforms using the proposed algorithms for (P9.1), (P9.2), and (P9.3), are seen to provide much better sparsification and recovery compared to the analytical DCT. Moreover, they typically perform comparably to the transforms learned via (P9.0). Importantly, the algorithm for (P9.0) has no convergence guarantee. Even the objective of (P9.0) is not guaranteed to converge for the algorithm in [2]. On an average, the rPSNR obtained via (P9.3) is only 0.14 dB worse than that obtained using (P9.0). Moreover, the rPSNR obtained via (P9.2) is only 0.22 dB worse (on an average) than that obtained using (P9.0).

The performance metrics for our algorithms for (P9.2) and (P9.3) are almost identical. Importantly, the DOSLAM algorithm for (P9.2) has guaranteed convergence. The transforms learned via the fully convex Problem (P9.1), although performing quite better (1.23 dB better in rPSNR on an average) than the DCT, are slightly worse compared to the transforms learned via the non-convex problems. The rPSNR obtained via the learned transforms from (P9.1) is about 0.58 dB worse on an average compared to those obtained using (P9.0).

The run times for the various doubly sparse learning algorithms were similar in our experiments. Our results thus indicate the usefulness of the proposed doubly sparse model $B = I + S$, with S approximately skew-symmetric. While we adapted the transforms to specific images here, one could also learn a transform from a class of images, and use such a 'global' transform to represent unrelated test images from that class (similar to [88], where Problem (P9.0) is employed for such a task). An idea of interest that we do not explore in this particular work is the learning of a collection of transforms, each

Image	P9.1	P9.2	P9.3	P9.0	DCT
Barbara	3.56	2.95	2.92	2.78	4.37
Brain	0.50	0.43	0.42	0.42	0.62
Cameraman	0.61	0.48	0.48	0.44	1.15
Lena	2.95	2.55	2.51	2.37	3.14
Birds	4.03	3.38	3.28	3.05	4.35
Boat	4.39	3.80	3.77	3.44	4.66

Table 9.1: NSE (in percentage) for various images obtained using our algorithms for (P9.1), (P9.2), and (P9.3) (transforms thresholded to sparsity of $0.25 \times n^2$), along with the corresponding NSE obtained using (P9.0) [2], and with the fixed transform Φ.

Image	P9.1	P9.2	P9.3	P9.0	DCT
Barbara	34.99	35.32	35.42	35.59	33.76
Brain	45.32	45.72	45.77	45.77	44.12
Cameraman	43.34	44.19	44.16	44.39	40.27
Lena	37.74	37.97	38.06	38.13	37.13
Birds	37.91	38.04	38.24	38.39	37.16
Boat	34.71	34.90	35.00	35.20	34.19

Table 9.2: rPSNR (in dB) for various images obtained using our algorithms for (P9.1), (P9.2), and (P9.3) (transforms thresholded to sparsity of $0.25 \times n^2$), along with the corresponding rPSNR obtained using (P9.0) [2], and with the fixed transform Φ.

adapted to a specific region/feature/texture of an image. Such a collection of transforms may be thought of as an overcomplete transform for the image.

While we demonstrated the usefulness of the proposed transform learning schemes for representing images, these techniques may also be useful in other applications such as (blind) denoising [9, 2], or blind compressed sensing [104]. A more detailed study of these applications is beyond the scope of this chapter, and will be explored elsewhere.

9.6 Conclusions

In this chapter, we presented the first convex sparsifying transform learning formulation, and an algorithm guaranteeing fast convergence to a global optimum. We also presented a 'partially non-convex' formulation by replacing the ℓ_1 sparsity penalty on the sparse code with an ℓ_0 constraint. In this case, the proposed algorithm has a local convergence property. In practice, it is also typically insensitive to initialization. For comparison, we also included a ‘fully non-convex’ version of our doubly sparse convex formulation by replacing the ℓ_1 penalties on both the sparse code and transform with ℓ_0 constraints. All our proposed formulations give rise to significantly sparse and well-conditioned transforms with much lower sparsification errors and recovery PSNRs than analytical transforms. Importantly, our convex formulation performs (in image representation) only slightly worse than its non-convex (i.e., the ‘partially non-convex’, and ‘fully non-convex’) variants. The learned transforms obtained via our proposed formulations here perform comparably or somewhat worse than those learned using the non-convex (non-guaranteed) schemes of Chapter 3 in our (preliminary) experiments. This may be because we exploit a different set of properties (skew-symmetry) here than in Chapter 3. Nevertheless, we would like to emphasize that the schemes proposed in this chapter such as the convex or ‘partially non-convex’ schemes have the advantage of strong convergence guarantees. The usefulness of the proposed transform learning schemes in image denoising or other applications merits detailed study.

CHAPTER 10

LEARNING UNSTRUCTURED OVERCOMPLETE SPARSIFYING TRANSFORMS

10.1 Introduction

In this chapter, we investigate the data-driven adaptation of (unstructured) overcomplete sparsifying transforms [1].

In the previous chapters, we considered various formulations and algorithms for square transform learning. The algorithms therein have a much lower computational cost compared to synthesis and analysis dictionary learning, and moreover, also provide convergence of the cost and iterates regardless of initial conditions. In this chapter, we however focus on the learning of unstructured *overcomplete* or tall sparsifying transforms, i.e., $W \in \mathbb{R}^{m \times n}$, with $m > n$. We empirically illustrate the convergence of our proposed learning algorithm, and demonstrate its usefulness in image denoising.

The rest of this chapter is organized as follows. Section 10.2 discusses our problem formulation for overcomplete transform learning and details the proposed algorithm and its properties. In Section 10.3, the convergence and learning behavior of our algorithm is illustrated using a numerical example. In Section 10.4, we demonstrate the usefulness of overcomplete transform learning in image denoising. In Section 10.5, we conclude.

[1]The material of this chapter has been previously presented in [172] and [173].

10.2 Formulation and Algorithm

10.2.1 Problem Formulation

Given a matrix $Y \in \mathbb{R}^{n \times N}$ whose columns represent training signals, the unstructured square transform learning Problem (P2.3) was presented in Chapter 2. We now extend (P2.3) to the overcomplete transform ($W \in \mathbb{R}^{m \times n}$, $m > n$) case. For the overcomplete or tall case, we replace $\log \det W$ in (P2.3) with $\log \det \left(W^T W\right)$, which would enable full column rank of W. Note that in this case, $\det \left(W^T W\right)$ is always non-negative. The $\log \det \left(W^T W\right)$ and $\|W\|_F^2$ penalties together help control the conditioning of the columns of W. However, good conditioning of $W^T W$ alone is not sufficient to ensure meaningful tall transforms. For instance, consider a tall W of the form

$$W = \begin{bmatrix} W_1 \\ 0_{m-n \times n} \end{bmatrix}$$

where W_1 is a well-conditioned square transform learned using (P2.3) and $0_{m-n \times n}$ is a matrix of zeros. In this case, $W^T W$ is well-conditioned, since $W^T W = W_1^T W_1$. Moreover, W is a candidate sparsifying 'tall' transform. However, such a tall W has the ambiguity of repeated zero rows and the penalty $\log \det \left(W^T W\right)$ is unable to preclude such a W.

Hence, we introduce an additional penalty $\sum_{j \neq k} |\langle w_j, w_k \rangle|^p$, that enforces incoherence between the rows of W, denoted as w_j ($1 \leq j \leq m$). The notation $\langle \cdot, \cdot \rangle$ stands for the standard inner product between vectors. Note that larger values of p emphasize the peak coherence. When $p = 2$, we can consider for example, a $W = \begin{bmatrix} W_1 \\ W_2 \end{bmatrix}$ that is a concatenation of two orthonormal transforms W_1 and W_2 (e.g., DCT and identity). The penalty $\sum_{j \neq k} \langle w_j, w_k \rangle^2$ is, however, a fixed constant when W consists of such orthonormal blocks, irrespective of the choice of those blocks. For this reason, we consider $p \gg 2$ (e.g., a large even natural number), to enforce better incoherence.

We also additionally constrain the rows of W to unit norm. Under this constraint, the penalty $|\langle w_j, w_k \rangle|$ truly measures the incoherence (or angle) between the rows w_j and w_k. Thus, our problem formulation for overcomplete transform learning is as follows.

$$\begin{aligned}(\text{P10.1})\ \min_{W,X}\ & \|WY - X\|_F^2 - \lambda \log\det\left(W^T W\right) \\ & + \eta \sum_{j\neq k} |\langle w_j, w_k\rangle|^p \\ s.t.\ \ & \|X_i\|_0 \leq s\ \ \forall\ i,\ \ \|w_k\|_2 = 1\ \ \forall\ k\end{aligned}$$

where $\eta > 0$ weights the incoherence penalty. Note that the $\|W\|_F^2$ penalty is a constant under the unit row norm assumption. Problem (P10.1) is however, non-convex.

10.2.2 Algorithm and Properties

Our algorithm for solving (P10.1) alternates between updating X and W. In one step called the **Sparse Coding Step**, we solve (P10.1) with fixed W as follows.

$$\min_{X}\ \|WY - X\|_F^2\ \ s.t.\ \ \|X_i\|_0 \leq s\ \ \forall\ i \tag{10.1}$$

The solution $\hat{X}$ is computed exactly by thresholding WY, and retaining the s largest coefficients (in magnitude) in each column. Note that if the l_0 "norm" for sparsity is relaxed to an l_1 norm and added as a penalty in the cost (10.1), we can still obtain an exact solution for X by soft thresholding [9].

In the second step of our algorithm called the **Transform Update Step**, we solve Problem (P10.1) with fixed X as follows.

$$\begin{aligned}\min_{W}\ & \|WY - X\|_F^2 - \lambda \log\det\left(W^T W\right) + \eta \sum_{j\neq k} |\langle w_j, w_k\rangle|^p \\ s.t.\ \ & \|w_k\|_2 = 1\ \ \forall\ k\end{aligned} \tag{10.2}$$

This problem does not have an analytical solution, and is moreover non-convex. We could solve for W using iterative algorithms such as the projected conjugate gradient method. However, we observed that the alternative strategy of employing the standard conjugate gradient (CG) algorithm, followed by post-normalization of the rows of W led to better empirical performance in applications. Hence, we choose the alternative strategy. When employing the standard CG, we also retain the $\|W\|_F^2$ penalty in the cost for CG, to prevent the scaling ambiguity [9].

The gradient expressions for the various terms in the cost (10.2) are as follows (cf. [84]). We choose p to be an even natural number (for simplicity), and assume $\det\left(W^T W\right) > 0$ on some neighborhood of W, otherwise the $\log(\cdot)$ function would be discontinuous.

$$\nabla_W \log\det\left(W^T W\right) = 2W\left(W^T W\right)^{-1} \tag{10.3}$$

$$\nabla_W \|WY - X\|_F^2 = 2WYY^T - 2XY^T \tag{10.4}$$

$$\nabla_W \sum_{j\neq k} |\langle w_j, w_k\rangle|^p = 2p\,(ZW - B) \tag{10.5}$$

The matrices $Z \in \mathbb{R}^{m\times m}$ and $B \in \mathbb{R}^{m\times n}$ above have entries $z_{ij} = \langle w_i, w_j\rangle^{p-1}$ and $b_{ij} = z_{ii} w_{ij}$.

The computational cost per iteration (of sparse update and transform update) of the proposed algorithm scales as $O(mnN)$ for learning an $m \times n$ transform from N training vectors. Note that this cost is typically much lower than the per-iteration cost of learning an $n \times K$ synthesis dictionary D using K-SVD [17], which scales as $O(Kn^2N)$ [5] (assuming that the synthesis sparsity $s \propto n$).

10.3 Convergence and Learning

In this section, we illustrate the convergence of our learning algorithm for (P10.1), and its ability to learn meaningful transforms. We learn a 128×64 transform from the 8×8 non-overlapping patches of the Barbara image [17]. The means of the patches are removed, and we only sparsify the mean-subtracted patches. We fix our algorithm parameters as $s = 11$, $p = 20$, $\lambda = \eta = 4 \times 10^5$. The CG algorithm in the transform update step is executed for 128 iterations with a fixed step size of 10^{-9}. Note that we use a weighting $\mu = \lambda$ for the Frobenius-norm penalty within CG. The algorithm is initialized with the (vertical) concatenation of the 2D DCT (obtained as the Kronecker product of two 8×8 1D DCT matrices) and identity matrices.

Figure 10.1 plots the objective function and sparsification error for our algorithm over iterations. Both the objective and sparsification error converge quickly. Moreover, the sparsification error improves significantly (by more than 8 dB) over the iterations compared to that of the initial transform (DCT concatenated with identity). We measure the normalized sparsification error [9] defined as $\|WY - X\|_F^2 / \|WY\|_F^2$. As discussed in previous chapters, this measures the fraction of energy lost in sparse fitting in the transform domain, which is an interesting property to observe for the adapted transforms. The normalized sparsification error for the final learned transform is 0.08, while that for the initialization is 0.42, indicating the significantly enhanced sparsification ability of the adapted transform.

The learned transform is shown in Figure 10.1, with each of its row displayed as an 8×8 patch, called the 'transform atom'. The atoms appear different from each other, and exhibit a lot of geometric and frequency like structures, which are reflective of the fact that the Barbara image has a lot of structure, textures. The learned transform is also well-conditioned with a condition number of 2.4. The magnitude of WW^T, shown in Figure 10.1, indicates mostly small values for the off-diagonal elements. The mutual coherence of W [174] (maximum off-diagonal magnitude in WW^T) is 0.88. The results here indicate the fast convergence of our algorithm, and its ability to learn meaningful, non-trivial overcomplete sparsifying transforms.

10.4 Image Denoising

Here, we consider the problem of recovering an image $x \in \mathbb{R}^P$ (2D image represented as vector) from its measurement $y = x + g$ corrupted by noise g. Our formulation for image denoising with adaptive overcomplete sparsifying transforms is as follows.

$$
\begin{aligned}
(P10.2)\ \min_{W, x_i, \alpha_i} & \sum_{i=1}^{M} \|Wx_i - \alpha_i\|_2^2 + \lambda Q(W) + \tau \sum_{i=1}^{M} \|R_i\, y - x_i\|_2^2 \\
s.t. \ \ & \|\alpha_i\|_0 \leq s_i \ \ \forall\ i \ , \ \|w_k\|_2 = 1 \ \ \forall\ k
\end{aligned}
$$

where $Q(W) = -\log \det \left(W^T W\right) + \frac{\eta}{\lambda} \sum_{j \neq k} |\langle w_j, w_k \rangle|^p$ represents the portion of the objective depending on only W. Operator $R_i \in \mathbb{R}^{n \times P}$ extracts a $\sqrt{n} \times \sqrt{n}$ patch from the noisy image y as $R_i y$ (we assume M overlapping patches). We model the noisy patch $R_i\, y$ as being approximated by a noiseless patch x_i, that is approximately sparsifiable in an adaptive *overcomplete* transform

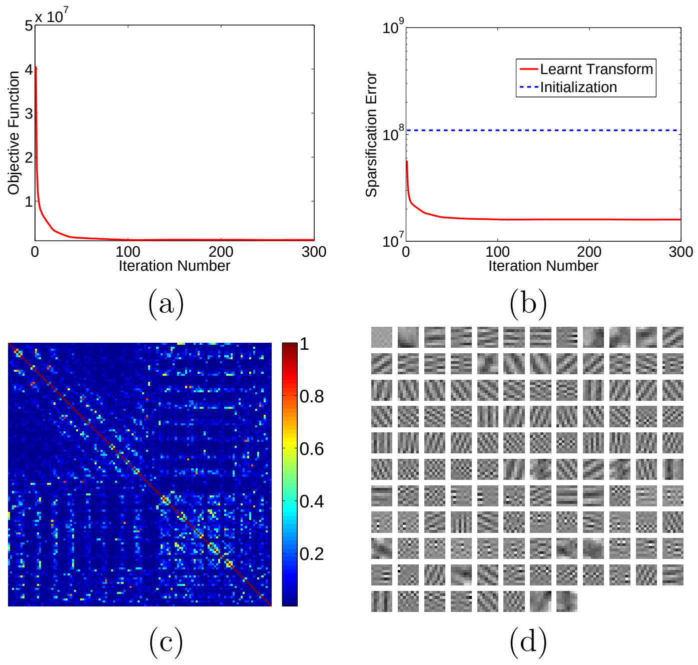


Figure 10.1: Convergence and Learning: (a) Objective function vs. iterations; (b) Sparsification error vs. iterations along with the sparsification error of the initial transform; (c) Magnitude of WW^T; and (d) Rows of learned transform shown as patches.

W. Vector $\alpha_i \in \mathbb{R}^n$ denotes the sparse code of x_i with s_i non-zeros. The weighting τ in (P10.2) is typically chosen as inversely proportional to the noise level σ [1].

Problem (P10.2) is non-convex. Similar to the algorithm for square transform-based image denoising (cf. Chapter 3), we propose a two-step iterative procedure to solve (P10.2). In the *transform learning* **Step 1**, we fix $x_i = R_i\, y$ and $s_i = s$ (fixed input s initially) in (P10.2), and solve for W and $\alpha_i\ \forall i$, using the proposed overcomplete transform learning algorithm. In the *variable sparsity update* **Step 2**, we update the sparsity levels s_i for all i. For fixed W and α_i, (P10.2) is a least squares problem, which can be solved independently for each x_i. However, we don't fix α_i, but rather only let it be a thresholded version of $WR_i\, y$ (since learning was done on $R_i\, y$), and adaptively find the sparsity level s_i.

The sparsity level s_i for the i^{th} patch needs to be chosen such that the denoising error term $\|R_i\, y - x_i\|_2^2$ computed after updating x_i by least squares (with α_i held at $H_{s_i}(WR_i y)$, where $H_{s_i}(\cdot)$ is the operator that retains the s_i

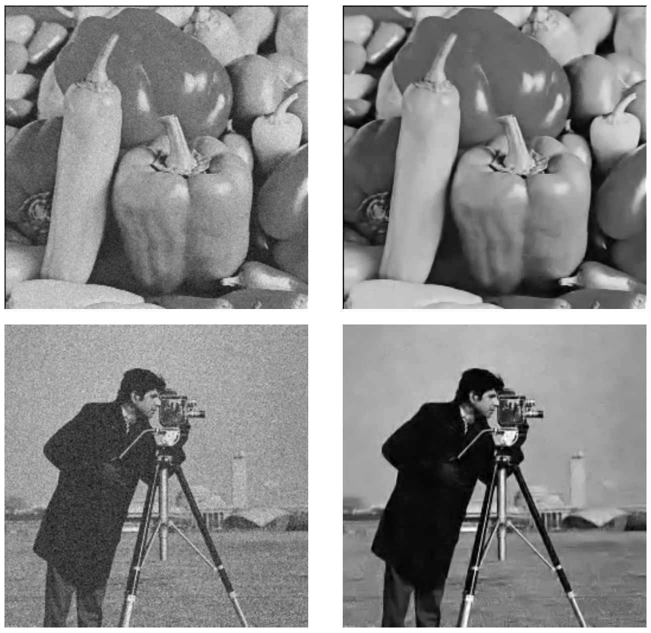


Figure 10.2: Noisy Images (left), Denoised Images (right).

components of largest magnitude in a vector, and sets the remaining components to zero) is below $nC^2\sigma^2$ [1], with C being a fixed parameter. Note that the denoising error term (with the updated x_i) decreases to zero, as $s_i \nearrow n$. Thus, finding s_i requires in general, repeating the least squares update of x_i for each i at various sparsity levels incrementally, to determine the level at which the error term falls below the required threshold. However, this process can be done very efficiently (cf. Chapter 3 for details).

Once the variable sparsity levels s_i are chosen for all i, we use the new s_i's back in the transform learning Step 1, and iterate over the learning and variable sparsity update steps, which leads to a better denoising performance compared to one iteration. In the final iteration, the x_i's that are computed (satisfying the $\|R_i\, y - x_i\|_2^2 \leq nC^2\sigma^2$ condition) represent the denoised patches.

Once the denoised patches x_i have been estimated, the denoised image x is obtained by averaging the x_i's at their respective locations in the image. The x is then restricted to its range (e.g., 0-255), if known. Note that we work with mean subtracted patches in our algorithm and typically learn on a subset of all patches (cf. [87]). The means are added back to the denoised patch estimates.

We now present some preliminary results for our overcomplete transform-based denoising framework. The goal here is to illustrate the potential for adaptive overcomplete transforms in this classical and prototypical application. We add i.i.d. Gaussian noise at noise level $\sigma = 10$ to the peppers image [1]. The denoising algorithm is executed for 3 iterations with parameters $n = 64$, $m = 100$, $\eta = \lambda = 8 \times 10^6$, $p = 20$, initial sparsity $s = 0.15 \times n$ (rounded to nearest integer), $C = 1.08$, and $\tau = 0.01/\sigma$. Transform learning is executed for 80 iterations (the weighting for the Frobenius-norm term within CG is λ).

The noisy image (PSNR = 28.1 dB) is shown along with its denoised version (PSNR = 34.49 dB) in Figure 10.2. The learned transform in this case has a condition number of 2.1 (well-conditioned), and also has incoherent rows (mutual coherence of 0.785). We compared our denoising performance to that obtained with the 64×256 K-SVD overcomplete synthesis dictionary [1, 86], which provided a lower denoising PSNR of 34.21 dB. Our denoising algorithm also takes less time (2.95 mins) compared to K-SVD (9.5 mins), due to the lower computational cost of sparse coding in the transform model. Note that we used a smaller training set for learning compared to K-SVD, since the 100×64 transform has fewer free parameters. The adapted overcomplete sparsifying transform also denoises better than the adapted square transform [9] learned using Problem (P2.3) (PSNR for the latter is 34.38 dB), indicating the usefulness of overcompleteness.

We also repeat the denoising experiment with overcomplete transforms for the cameraman image using a high noise of $\sigma = 20$. The noisy image (PSNR = 22.1 dB), and its denoised version (PSNR = 29.95 dB) are shown in Figure 10.2. The denoising PSNR obtained using adaptive overcomplete transforms is better than that obtained using the 64×256 K-SVD synthesis dictionary (PSNR = 29.84 dB) [1, 86]. Our denoising algorithm is also 7x faster than K-SVD [1]. (Note that we used a smaller number of 20 learning iterations here, to prevent overfitting to high noise.) We expect the run times for our algorithm to decrease substantially with code optimization.

We expect the denoising performance of our algorithms to improve/become comparable to the state of the art (for example [175]) with better choice of parameters, and with further extensions of overcomplete transform learning (e.g., multiscale transforms).

10.5 Conclusions

In this chapter, we introduced a novel problem formulation for learning overcomplete sparsifying transforms. The proposed alternating algorithm for overcomplete transform learning involves a sparse coding step and a transform update step. The solution of the sparse coding step is cheap and exact, and we use iterative methods (CG) for the transform update step. The learned transforms have better properties compared to the initialization. Moreover, the computational cost of overcomplete transform learning is lower than that of overcomplete dictionary learning. We also applied the adaptive overcomplete sparsifying transforms to image denoising, where they provide better performance over the synthesis K-SVD, while being faster. The overcomplete transforms also denoise better than square transforms.

CHAPTER 11

STRUCTURED OVERCOMPLETE SPARSIFYING TRANSFORM LEARNING

11.1 Introduction

A drawback of many of the prior transform learning methods is that they are restricted to the case of square transforms. For natural images with highly complicated structures, a single learned square transform may not provide sufficient sparsification. Hence, in this chapter [1], we present a union of sparsifying transforms model, and show that it can represent the diverse features, or textures seen in natural images much more accurately than a single sparsifying transform. In applications, the learning of this model also turns out to be more advantageous than prior (sparse) learning-based approaches. In the following, we further discuss the highlights of this work.

11.1.1 Our Contributions

11.1.1.1 Structured Overcomplete Transform Model

We investigate a union of square sparsifying transforms model in this work. In this model, we consider a collection (union) of square transforms $\{W_i\}_{i=1}^{K}$. A candidate signal is said to match (or, belong to) a particular transform in the collection if that transform provides the *best sparsification* for the signal among all the transforms in the collection.

A motivation for the proposed model is that natural signals and images (even if they belong to a single class such as MRI images, music signals, etc.) need not be sufficiently sparsifiable by a single transform. For example, image patches from different regions of a natural image usually contain different

[1] This is a joint work with B. Wen (equal contributor), University of Illinois. The material of this chapter appears in [134] and [176].

features, or textures. Thus, having a union of transforms would allow groups of patches with common features (or, textures) to be better sparsified by their own texture-specific transform.

We will show that this union of square transforms model can be interpreted as an overcomplete sparsifying transform model with an additional constraint of block cosparsity for the transform sparse code. Here, the overcomplete transform is formed by stacking the transforms in $\{W_i\}_{i=1}^{K}$ on top of each other. For the sake of brevity, we will also refer to our OverComplete TransfOrm model with BlOck coSparsity constraint as the OCTOBOS model. In the remainder of this chapter, we will use the terms 'union of transforms', or 'OCTOBOS' interchangeably, depending on the context.

11.1.1.2 Highlights

We enumerate some important features of our work as follows.

(i) Sparse coding in the proposed OCTOBOS model reduces to a form of clustering and is computationally inexpensive.

(ii) In this work, we propose a novel problem formulation and algorithm for learning structured overcomplete sparsifying transforms with block cosparsity constraint. Our algorithm is an alternating minimization algorithm, and each step of the algorithm involves simple (computationally cheap) closed-form solutions.

(iii) We present a novel convergence guarantee for the proposed alternating OCTOBOS learning algorithm. We prove global convergence (i.e., convergence from any initialization) of the algorithm to the set of partial minimizers of the objective defining the problem formulation. We also show that under certain conditions, the algorithm converges to the set of stationary points of the overall objective.

(iv) Our adapted OCTOBOS model provides a better sparse representation of images than adaptive single square transforms and analytical transforms such as the DCT.

(v) We present an adaptive image denoising formulation and algorithm exploiting the OCTOBOS model in this work. The denoising performance of the proposed approach is better than that obtained using adaptive square transforms, or adaptive overcomplete synthesis dictionaries (K-SVD). Our denoising scheme also performs better than the well-known Gaussian Mixture Model (GMM) approach [8], and is comparable to the state-of-the-art BM3D denoising [7] in some cases.

11.1.1.3 Related Work

A model similar to the union of transforms, but involving instead a union of orthogonal synthesis dictionaries (PCAs) has been recently used by Peleg and Elad [177] for the task of single image super-resolution. Also, for the specific task of super-resolution, Wang et al. [178] learn a union of coupled synthesis dictionaries.

The learning of a union of synthesis dictionaries with the main goal of unsupervised classification has been previously proposed in a number of works [179, 180, 181]. The learning of structured synthesis dictionary models (with block, or group sparsity) for tasks such as classification has also been explored [182, 183, 184, 185].

Similar to prior work on dictionary learning, these various formulations tend to be highly non-convex, and these approaches suffer from the high computational cost associated with sparse coding in the synthesis model. In contrast, eliminating the NP hard sparse coding or its approximation, our proposed OCTOBOS learning scheme has a low computational cost.

In this work, we only briefly discuss the possibility of classification (or, segmentation) using our proposed transform learning scheme. Indeed, the classification application is not the focus of this work, and a detailed study of this application will be considered for future work. Instead, to illustrate the usefulness of the learned union of transforms/OCTOBOS model, we focus in this work on applications such as image representation and denoising. We also provide convergence guarantees for OCTOBOS learning. Such guarantees are not available for the methods that learn a union of synthesis dictionaries, or block-sparse synthesis models.

11.1.1.4 Organization

The proposed union of square transforms model and its various alternative interpretations are described in Section 11.2. Section 11.2 also introduces our newly proposed learning formulation. In Section 11.3, we describe our algorithm for learning the proposed structured overcomplete sparsifying transform, and discuss the algorithm's computational properties. In Section 11.4, we provide a convergence analysis for our transform learning algorithm. The application of our transform learning framework to image denoising is discussed in Section 11.5. We then present experimental results demonstrating the convergence behavior, and promising performance of our proposed approach in Section 11.6. In Section 11.7, we conclude.

11.2 OCTOBOS Model and Learning Formulation

11.2.1 The Union of Transforms Model

The square sparsifying transform model has been investigated [9] recently. Here, we extend the single square transform model to a union of transforms model, which suggests that a signal $y \in \mathbb{R}^n$ is approximately sparsifiable by a particular transform in the collection $\{W_k\}_{k=1}^K$, where $W_k \in \mathbb{R}^{n\times n}\,\forall\, k$ are themselves square transforms. Thus, there exists a particular W_k such that $W_k y = x + e$, with $x \in \mathbb{R}^n$ sparse, and a transform residual e that is sufficiently small.

Given a signal $y \in \mathbb{R}^n$, and a union (or, collection) of square transforms $\{W_k\}_{k=1}^K$, we need to find the best matching transform (or, model) for the signal, that gives the smallest sparsification error. This can be formulated as the following sparse coding problem:

$$\text{(P11.1)}\quad \min_{1\leq k\leq K}\ \min_{z^k}\ \left\|W_k y - z^k\right\|_2^2 \quad s.t.\ \left\|z^k\right\|_0 \leq s\ \ \forall\ k$$

Here, z^k denotes the sparse representation of y in the transform W_k, with the maximum allowed sparsity level being s. We assume that the W_k's are all identically scaled in (P11.1). Otherwise, they can be rescaled (for example, to unit spectral or Frobenius norm) prior to solving (P11.1).

In order to solve (P11.1), we first find the optimal sparse code $\hat{z}^k$ for each[2] k as $\hat{z}^k = H_s(W_k y)$, where the operator $H_s(\cdot)$ is the projector onto the s-ℓ_0 ball, i.e., $H_s(b)$ zeros out all but the s elements of largest magnitude in $b \in \mathbb{R}^n$. If there is more than one choice for the s coefficients of largest magnitude in a vector b, which can occur when multiple entries in b have identical magnitude, then we choose $H_s(b)$ as the projection of b for which the indices of the s largest magnitude elements in b are the lowest possible. Now, Problem (P11.1) reduces to

$$\min_{1 \leq k \leq K} \|W_k y - H_s(W_k y)\|_2^2 \tag{11.1}$$

To solve the above problem, we compute the sparsification error (using the optimal sparse code above) for each k and choose the best transform $W_{\hat{k}}$ as the one that provides the smallest sparsification error (among all the W_k's). This is an exact solution technique for Problem (P11.1). Problem (P11.1) then also provides us with an optimal sparse code $\hat{z}^{\hat{k}} = H_s(W_{\hat{k}} y)$ for y. Given such a sparse code, one can also recover a signal estimate by minimizing $\left\| W_{\hat{k}} y - \hat{z}^{\hat{k}} \right\|_2^2$ over all $y \in \mathbb{R}^n$. The recovered signal is then given by $\hat{y} = W_{\hat{k}}^{-1} \hat{z}^{\hat{k}}$.

Since Problem (P11.1) matches a given signal y to a particular transform, it can be potentially used to cluster a collection of signals according to their transform models. The sparsification error term in (11.1) can be viewed as a clustering measure in this setting. This interpretation of (P11.1) indicates the possible usefulness of the union of transforms model in applications such as classification.

11.2.2 The Overcomplete Transform Model Interpretation

We now propose an interpretation of the union of transforms model as a structured overcomplete transform model (or, the OCTOBOS model). The 'equivalent' overcomplete transform is obtained from the union of transforms by stacking the square sub-transforms as $W = \left[W_1^T \mid W_2^T \mid \; ... \; \mid W_K^T \right]^T$. The tall matrix $W \in \mathbb{R}^{m \times n}$, with $m = Kn$, and thus, $m > n$ (overcomplete

[2]For each k, this is identical to the single transform sparse coding problem.

transform) for $K > 1$.

The signal y is assumed to obey the model $Wy = x + e$, where the $x \in \mathbb{R}^m$ is assumed to be "block cosparse", and e is a small residual. The block cosparsity of x is defined here using the following ℓ_0-type norm:

$$\|x\|_{0,s} = \sum_{k=1}^{K} I(\|x^k\|_0 \leq s) \tag{11.2}$$

Here, $x^k \in \mathbb{R}^n$ is the block of x corresponding to the transform W_k in the tall W, and s is a given sparsity level (equivalently $n - s$ is the given cosparsity level) for block x^k. The operator $I(\cdot)$ above is an indicator function with $I(Q) = 1$ when statement Q is true, and $I(Q) = 0$ otherwise. We say that x is 1-block cosparse if there is exactly one block of x with at least $n - s$ zeros, i.e., $\|x\|_{0,s} = 1$ in this case.

In the proposed overcomplete transform model for signal y, we formulate the following sparse coding problem, to which we refer as the OCTOBOS sparse coding problem.

$$\text{(P11.2)} \quad \min_{x} \ \|Wy - x\|_2^2 \quad s.t. \quad \|x\|_{0,s} \geq 1$$

Problem (P11.2) finds an x with at least one block that has $\geq n - s$ zeros. In particular, we now prove the following proposition, that the Problems (P11.1) and (P11.2) are equivalent. This equivalence is the basis for the interpretation of the union of transforms model as an overcomplete transform model.

Proposition 14. *The minimum values of the sparsification errors in Problems (P11.1) and (P11.2) are identical. The optimal sparse code(s) in (P11.1) is equal to the block(s) of the optimal $\hat{x}$ in (P11.2) satisfying $\|\hat{x}^k\|_0 \leq s$.*

Proof. : The objective in (P11.2) is $\sum_{k=1}^{K} \|W_k y - x^k\|_2^2$. The constraint in (P11.2) calls for $\|x^k\|_0 \leq s$ for at least one k. Assume without loss of generality that in the optimal solution $\hat{x}$ of (P11.2), the blocks $\hat{x}^k$ satisfying the constraint $\|\hat{x}^k\|_0 \leq s$ have indices $1, ..., J$ (for some $J \geq 1$). Otherwise, we can always trivially permute the blocks $\hat{x}^k$ in $\hat{x}$ (and the corresponding W_k in W) so that the optimal indices are $1, ..., J$. Now, for fixed optimal block indices, Problem (P11.2) reduces to the following Problem.

$$\begin{aligned} \min_{\{x^k\}} \; & \sum_{k=1}^{K} \left\| W_k y - x^k \right\|_2^2 \\ s.t. \; & \left\| x^k \right\|_0 \leq s, \text{ for } k = 1, ..., J \end{aligned} \tag{11.3}$$

Here, the notation $\{x^k\}$ denotes the set of x^k for $1 \leq k \leq K$ [3]. Since the blocks with indices $k > J$ are not selected by the sparsity constraint, for $k > J$ the optimal $\hat{x}^k = W_k y$, because this setting results in a zero (minimal) contribution to the objective (11.3) above. Problem (11.3) then decouples into J independent (square) transform sparse coding problems. It is then obvious that the minimum in (11.3) is achieved with $J = 1$ (minimum possible J), i.e., we only choose one block as $\hat{x}^k = H_s(W_k y)$, and all other blocks satisfy $\hat{x}^k = W_k y$. This setting leads to a zero contribution to the objective of (11.3) for all but one block. The chosen active block is the one that provides the smallest (minimizes (11.3)) individual sparsification error. It is now obvious (by directly comparing to the sparse coding algorithm for (P11.1)) that the proposition is true.

Note that if $\|W_k y\|_0 \leq s$ holds for one or more k, then the optimal $\hat{x} = Wy$ in (P11.2). Only in this degenerate case, it is possible for the optimal $\hat{x}$ in (P11.2) to have more than one block that is s-sparse (i.e., $\|\hat{x}\|_{0,s} > 1$ occurs if $\|W_k y\|_0 \leq s$ for two or more k). In this case, the optimal sparse code in (P11.1) can be set to be equal to any of the optimal s-sparse blocks in $\hat{x} = Wy$. The minimum sparsification error is zero for both (P11.1) and (P11.2) in this degenerate case. $\square$

The optimal $\hat{x}$ in (P11.2) by itself cannot be called a sparse code, since (based on the proof of Proposition 14) it typically has many more non-zeros than zeros [4]. However, the particular s-sparse block(s) of $\hat{x}$ can be considered as a sparse code, and one could also recover a signal estimate from this code similar to the union of transforms case [5]. Note that the many non-zeros in

[3]In the remainder of the chapter, when certain indexed variables are enclosed within braces, it means that we are considering the set of variables over the range of all the indices.

[4]For example, when vector Wy has no zeros, then the optimal $\hat{x}$ in (P11.2) has exactly $n - s \ll Kn$ (for large K) zeros – all the zeros are concentrated in a single block of $\hat{x}$.

[5]More precisely, the index of the sparse block is also part of the sparse code. This adds just $\log_2 K$ bits per index to the sparse code.

$\hat{x}$ help keep the overcomplete transform residual small.

The OCTOBOS model enforces a block cosparsity constraint. Alternatively, one could consider the model $Wy = x + e$ with a tall transform $W \in \mathbb{R}^{Kn\times n}$, but without any block cosparsity constraint on x, and assuming that x has at least $n-s$ zeros, i.e., $\|x\|_0 \leq (K-1)n + s$. The sparse coding in this model would be identical to thresholding (zeroing out the $n-s$ elements of smallest magnitude of) Wy. However, it is unclear how to easily combine the non-zeros and zeros to form a length n sparse code [6]. Therefore, we do not pursue this case (non-block cosparse model) in this work.

11.2.3 An OCTOBOS Optimality Property

Here, we consider two data matrices $Y_1 \in \mathbb{R}^{n\times N}$ and $Y_2 \in \mathbb{R}^{n\times M}$ (columns of the matrices represent signals), each of which is sparsified by a different square transform. We provide a condition under which using just one of the two transforms for both Y_1 and Y_2 will increase the total sparsification error (computed over all signals in Y_1 and Y_2). Thus, when the proposed condition holds, the union of transforms provides a better model for the collection of data compared to any one transform.

The proposed condition is based on the spark property [186]. For a matrix $A \in \mathbb{R}^{n\times r}$, the spark is defined to be the minimum number of columns of A that are linearly dependent.

Proposition 15. *Given two sets of data $Y_1 \in \mathbb{R}^{n\times N}$ and $Y_2 \in \mathbb{R}^{n\times M}$, suppose there exist non-identical and non-singular square transforms $W_1, W_2 \in \mathbb{R}^{n\times n}$, that exactly sparsify the datasets as $W_1 Y = X_1$ and $W_2 Y_2 = X_2$, where the columns of both X_1 and X_2 have sparsity $\leq s$. If* spark $\left[W_1^{-1} \mid W_2^{-1}\right] > 2s$, *then the columns of $W_2 Y_1$ have sparsity $> s$.*

Proof. : Consider an arbitrary column, say the ith one, of Y_1, which we denote as z. Let $\alpha_i^1 = W_1 z$. We then have that $\|\alpha_i^1\|_0 \leq s$. Let us denote $W_2 z$ by α_i^2. We then have that

$$\left[W_1^{-1} \mid W_2^{-1}\right] \begin{bmatrix} \alpha_i^1 \\ -\alpha_i^2 \end{bmatrix} = B\alpha_i = 0 \tag{11.4}$$

[6]We need a length n code in a square and invertible sub-transform of W, in order to perform signal recovery uniquely.

where $B = \left[W_1^{-1} \mid W_2^{-1}\right]$, and α_i is the vertical concatenation of α_i^1 and $-\alpha_i^2$. Now, if matrix B has spark $> 2s$, then the linear combination of any $\leq 2s$ of its columns cannot equal zero. Therefore, under the spark assumption, we must have $\|\alpha_i\|_0 > 2s$. Since, $\|\alpha_i^1\|_0 \leq s$, we must then have $\|\alpha_i^2\|_0 > s$, under the spark assumption. $\square$

If the spark condition above holds, then the sparsification errors of the columns of Y in W_2 (using sparsity level s) are strictly positive. We can also derive an alternative condition that involves the mutual coherence of $B = \left[W_1^{-1} \mid W_2^{-1}\right]$. The mutual coherence of the matrix B [27] is defined as follows.

$$\mu(B) = \max_{1 \leq k,j \leq m, k \neq j} \frac{\left|B_k^T B_j\right|}{\|B_k\|_2 \cdot \|B_j\|_2} \tag{11.5}$$

Unlike the spark, the mutual coherence is easy to compute, and characterizes the dependence between the columns (indexed by the subscripts j and k in (11.5)) of matrix B. It is known that the spark and mutual coherence of a matrix B are related as follows [27].

$$\text{spark}(B) \geq 1 + \frac{1}{\mu(B)} \tag{11.6}$$

Therefore, in Proposition 15, the spark condition can be replaced by the following (more stringent) sufficient condition involving the mutual coherence of B.

$$\mu(B) < \frac{1}{2s - 1} \tag{11.7}$$

If the above condition holds, then by equation (11.6), the spark condition of Proposition 15 automatically holds, and thus we will have that the columns of $W_2 Y_1$ have sparsity $> s$.

The spark-based sufficient condition in Proposition 15 can be interpreted as a similarity measure between the models W_1 and W_2. In the extreme case, when $W_1 = W_2$, the aforementioned matrix B has minimum possible spark $(= 2)$. In broad terms, if W_1 and W_2 are sufficiently different, as measured by the spark condition in Proposition 15, or the coherence condition in (11.7), then the union of transforms model, or OCTOBOS provides a better model than either one of the transforms alone.

The difference between W_1 and W_2 as measured by the spark, or coherence conditions is invariant to certain transformations. In particular, if W_1 is an exact full rank sparsifier of matrix Y_1, then one can also obtain equivalent transforms by permuting the rows of W_1, or by pre-multiplying W_1 with a diagonal matrix with non-zero diagonal entries. All these equivalent transforms sparsify Y_1 equally (i.e., provide the same sparsity level of s) well. It is easy to see that if the condition spark $\left[W_1^{-1} \mid W_2^{-1}\right] > 2s$ (or, alternatively, the mutual coherence-based condition) holds with respect to a particular W_1 and W_2 in Proposition 15, then it also automatically holds with respect to any other equivalent W_1 and equivalent W_2.

11.2.4 OCTOBOS Learning Formulations and Properties

11.2.4.1 Problem Formulations

Given the matrix $Y \in \mathbb{R}^{n\times N}$, whose columns represent training signals, recall that the unstructured square transform learning problem proposed in Chapter 4 is

$$\text{(P11.3)} \quad \min_{W,X} \; \|WY - X\|_F^2 + \lambda Q(W)$$
$$s.t. \;\; \|X_i\|_0 \leq s \;\; \forall \; i$$

where $Q(W) = -\log|\det W| + \|W\|_F^2$ here. The subscript i above denotes the ith column of the sparse code matrix X, and $W \in \mathbb{R}^{n\times n}$.

Similar to the square sparsifying transform learning problem, we now propose the following OCTOBOS learning formulation that learns a tall sparsifying transform $W \in \mathbb{R}^{Kn\times n}$ and sparse code matrix $X \in \mathbb{R}^{Kn\times N}$ from training data $Y \in \mathbb{R}^{n\times N}$.

$$\text{(P11.4)} \quad \min_{W,X} \; \|WY - X\|_F^2 + Q'(W)$$
$$s.t. \;\; \|X_i\|_{0,s} \geq 1 \;\; \forall \; i$$

Here, $\|X_i\|_{0,s}$ is defined as in equation (11.2). The function $Q'(W)$ is defined as follows.

$$Q'(W) = \sum_{k=1}^{K} \lambda_k Q(W_k) \tag{11.8}$$

The regularizer $Q'(W)$ controls the condition number of the sub-blocks of W, and λ_k are positive weights.

One can also formulate the transform learning problem in the union of transforms model as follows.

$$\text{(P11.5)} \min_{\{W_k, X_i, C_k\}} \sum_{k=1}^{K} \left\{ \sum_{i \in C_k} \|W_k Y_i - X_i\|_2^2 + \lambda_k Q(W_k) \right\}$$
$$s.t. \;\; \|X_i\|_0 \leq s \;\; \forall \; i, \;\; \{C_k\} \in G$$

Here, $X_i \in \mathbb{R}^{n \times n}$ and the set $\{C_k\}_{k=1}^{K}$ indicates a clustering of the training signals $\{Y_i\}_{i=1}^{N}$. The cluster C_k contains the indices i corresponding to the signals Y_i in the k^{th} cluster. The signals in the k^{th} cluster are matched to transform W_k. The set G is the set of all possible partitions of the set of integers $[1:N] \triangleq \{1, 2, ..., N\}$, or in other words, G is the set of all possible $\{C_k\}$, and is defined as follows.

$$G = \left\{ \{C_k\} : \bigcup_{k=1}^{K} C_k = [1:N], \, C_j \bigcap C_k = \emptyset, \, \forall \, j \neq k \right\}$$

The constraint involving G thus enforces the various C_k in $\{C_k\}_{k=1}^{K}$ to be disjoint, and their union to contain the indices for all training signals. Note that the term $\sum_{k=1}^{K} \sum_{i \in C_k} \|W_k Y_i - X_i\|_2^2$ in (P11.5) is the sparsification error for the data Y in the union of transforms model.

The weights λ_k is (P11.4) and (P11.5) are chosen as $\lambda_k = \lambda_0 \|Y_{C_k}\|_F^2$, where Y_{C_k} is a matrix whose columns are the signals of Y in the k^{th} cluster. The rationale for this choice of λ_k is similar to that presented previously in Chapter 4 for the λ weight in (P11.3). Specifically, when the clusters $\{C_k\}_{k=1}^{K}$ are fixed to their optimal values in (P11.5), the optimization problem (P11.5) reduces to K square transform learning problems of the form of (P11.3), each involving a particular data matrix Y_{C_k}. Thus, the setting $\lambda_k = \lambda_0 \|Y_{C_k}\|_F^2$ achieves scale invariance for the solutions of these K problems. The setting also implies that λ_k itself is a function of the unknown C_k (function of the signal energy in cluster C_k) in the optimization problem (P11.5). When the $\{W_k\}$ are fixed, the $Q'(W)$ penalty in (P11.5) encourages a larger concentration of data energy ($\|Y_{C_k}\|_F^2$) in the cluster corresponding to a smaller $Q(W_k)$ (i.e., corresponding to smaller condition number and reasonable scaling).

11.2.4.2 Properties of Formulations (P11.4) and (P11.5)

The following result implies that the learning Problems (P11.4) and (P11.5) are equivalent.

Proposition 16. *The minimum values of the objectives in Problems (P11.4) and (P11.5) are identical. Moreover, any optimal union of transforms in (P11.5) can be vertically concatenated to form an optimal overcomplete W for (P11.4). Similarly, any optimal W in (P11.4) can be used to generate an optimal union of transforms for (P11.5).*

Proof. : Let $\{W_k, C_k, X_{C_k}\}$ be a global minimizer of (P11.5). Then, we can form an equivalent overcomplete W by vertically stacking the W_k's. Moreover, we can form/construct a tall sparse code matrix $X' \in \mathbb{R}^{Kn \times n}$ by letting each column X'_i be equal to X_i on one block (according to the clustering C_k), and equal to $W_k Y_i$ on the other blocks. The constructed (W, X') is feasible for (P11.4), and provides a value for the (P11.4) objective that is equal to the minimum objective value attained in (P11.5). Thus, the minimum objective value in (P11.4) can only be lower than the minimum value in (P11.5).

Similarly, given an optimal minimizer (W, X) for (P11.4), we can form $\{W_k\}$ as the blocks of W. The $\{C_k\}$ and $\{X_i\}$ parts of (P11.5) can also be constructed from X using Proposition 14. The constructed $\{W_k\}$, $\{C_k\}$, $\{X_i\}$ is feasible for (P11.5), and provides a value for the (P11.5) objective that is clearly equal to the minimum objective value obtained in (P11.4). Since, the minimum in (P11.5) is computed over all feasible $\{W_k\}$, $\{C_k\}$, $\{X_i\}$, it can be only lower than the minimum objective value in (P11.4).

By the preceding arguments, it is clear that the minimum values of the objectives in (P11.4) and (P11.5) must in fact, be identical. The rest of the proposition also follows from the above arguments and construction techniques. $\square$

Although (P11.4) and (P11.5) are equivalent, Problem (P11.5) is more intuitive and amenable to alternating optimization schemes (see Section 11.3 for such a scheme). If we were to alternate between updating X and W in (P11.4), we would not be able to directly maintain (without additional constraints) the property that the transform domain residual for each Y_i is zero in all but (at most) one block of W, during the update of W. This is not a problem for (P11.5), since its objective only considers the residual of each Y_i in one (best) block.

The following result indicates that the minimum objective in the union of transforms Problem (P11.5) is always lower than the minimum objective value in the single transform learning problem (P11.3). This means that either the optimal sparsification error in (P11.5) is lower than the corresponding value in (P11.3), or the optimal regularizer (that controls the condition number(s) of the transform block(s)) in (P11.5) is smaller than the corresponding value in (P11.3), or both of these conditions hold.

Proposition 17. *The minimum value of the objective in (P11.5) can only be lower than the minimum objective value in (P11.3).*

Proof. : Let $(\hat{W}, \hat{X})$ denote an optimal minimizer of (P11.3), i.e., it provides the minimum value of the objective of (P11.3). Now, in (P11.5), we can set $W_k = \hat{W}\ \forall\, k$, and $X_i = \hat{X}_i\ \forall\, i$ in the objective. For this setting, the objective in (P11.5) becomes identical (using the fact that $\sum_{k=1}^{K} \lambda_k = \lambda$) to the minimum value of the objective in (P11.3). This result is invariant to the specific choice of the C_k's. Now, since the minimum value of the objective in (P11.5) is attained over all feasible $\{W_k\}$, $\{X_i\}$, $\{C_k\}$, it can only be lower ($\leq$) than the value obtained with the specific settings above. $\square$

We will empirically illustrate in Section 11.6 that our algorithm for (P11.5) (discussed in Section 11.3) provides a lower value of both the objective and sparsification error compared to the algorithm for (P11.3).

It has been shown [9] that the objective of Problem (P11.3) is lower bounded. The following lemma confirms that the objectives of the proposed learning formulations are lower bounded too.

Lemma 6. *The objectives in Problems (P11.4) and (P11.5) are both lower bounded by* $\lambda Q_0 = \lambda_0\, Q_0\, \|Y\|_F^2$*, where* $Q_0 = \frac{n}{2} + \frac{n}{2}\log(2)$*.*

Proof. : The objectives in (P11.4) and (P11.5) are the summation of a sparsification error term (net error over all signals) and a $Q'(W) = \sum_{k=1}^{K} \lambda_k Q(W_k)$ regularizer term. The sparsification error term is lower bounded by 0. Each $Q(W_k)$ regularizer is bounded as $Q(W_k) \geq Q_0 = \frac{n}{2} + \frac{n}{2}\log(2)$ (cf. [9] for proof of this). Thus, $Q'(W) \geq Q_0 \sum_{k=1}^{K} \lambda_k = \lambda_0\, Q_0\, \|Y\|_F^2$, where we used the setting $\lambda_k = \lambda_0\, \|Y_{C_k}\|_F^2$. Thus, the objectives in (P11.4) and (P11.5) are both lower bounded by $\lambda Q_0 = \lambda_0\, Q_0\, \|Y\|_F^2$. $\square$

We use the preceding lemma to prove the following proposition, which pertains to the identifiability of good models (models that sparsify well and are well-conditioned) by our Problem (P11.5). Proposition 18 also pertains to the case when the lower bounds in Lemma 6 are achievable.

Proposition 18. *Given a training data matrix $Y \in \mathbb{R}^{n \times N}$, let $\{Y_{C_k}\}$ be a collection of data matrices formed according to a clustering rule $\{C_k\}$. Suppose that $\{W_k\}$ is a collection of unit conditioned square transform models, each with spectral norm $1/\sqrt{2}$ [7], that exactly sparsifies the clustered data $\{Y_{C_k}\}$ as $W_k Y_{C_k} = X_{C_k}\ \forall\, k$, with each X_{C_k} having s-sparse columns. Then, the set $\{W_k, C_k, X_{C_k}\}$ is a global minimizer of (P11.5), i.e., the underlying model is identifiable by solving (P11.5).*

Proof. : The objective in (P11.5) is the summation of a sparsification error term and a $Q'(W)$ regularizer term. Since, $\{W_k\}$ exactly sparsify the clustered data $\{Y_{C_k}\}$, the sparsification error in (P11.5) is zero (minimum) at the given $\{W_k, C_k, X_{C_k}\}$. The regularizer term $Q'(W) = \sum_{k=1}^{K} \lambda_k Q(W_k)$ only depends on the $\{W_k\}$. It was recently shown [9] that $Q(W_k) \geq Q_0$, with equality if and only if W_k is unit conditioned, and the singular values of W_k are all equal to $\sqrt{\frac{1}{2}}$. Since, each W_k considered here achieves $Q(W_k) = Q_0$, we have that the regularizer $Q'(W)$ attains its lower bound $\lambda Q_0 = \lambda_0 Q_0 \|Y\|_F^2$ mentioned in Lemma 6, for the considered $\{W_k\}$. Thus, we have shown that the objective in (P11.5) attains its lower bound for the given $\{W_k, C_k, X_{C_k}\}$. In other words, the objective attains its global minimum in this case. □

Thus, when an "error-free" union of transforms model exists for the data, and the transforms are all unit conditioned, Proposition 18 guarantees that such a union of transforms model is a global minimizer of the proposed Problem (P11.5). Therefore, it makes sense to solve (P11.5) in order to find such good OCTOBOS models.

We now show that the role of the λ_0 weight in (P11.5) is to control the condition number and scaling of the transform blocks W_k ($1 \leq k \leq K$). If we were to minimize only the $\hat{Q}(W) = Q'(W)/\lambda_0 = \sum_{k=1}^{K} \|Y_{C_k}\|_F^2 \, Q(W_k)$ regularizer in Problem (P11.5) with respect to the unknowns, then the minimum value would be $Q_0 \|Y\|_F^2$ according to Lemma 6. This minimum is

[7]If the transforms have a different spectral norm, they can be trivially scaled to have spectral norm $1/\sqrt{2}$.

achieved with W_k's that are unit conditioned, and with spectral norm of $1/\sqrt{2}$ (i.e., transforms with identical scaling). Thus, similar to Corollary 2 in [9], we have that as $\lambda_0 \to \infty$ in (P11.5), the condition number of the optimal transforms in (P11.5) tends to 1, and their spectral norm (scaling) tends to $1/\sqrt{2}$. Therefore, as $\lambda_0 \to \infty$, our formulation (P11.5) approaches a union of unit-conditioned transforms learning problem. We also empirically show in Section 11.6 that when λ_0 is properly chosen (but finite), the condition numbers and norms of the learned W_k's in (P11.5) are very similar. Note that we need the W_k's to be similarly scaled for the sparsification error in (P11.5) to be fully meaningful (since otherwise, a certain W_k with a very small scaling can trivially give the best sparsification error for a signal).

Another interesting fact about OCTOBOS learning is that both (P11.4) and (P11.5) admit an equivalence class of solutions similar to (P11.3). For example, one can permute the rows within an optimal block W_k (along with a permutation of the corresponding sparse codes), or pre-multiply W_k by a diagonal ± 1 sign matrix (and multiply the sparse codes accordingly), without affecting its optimality. In (P11.4), one can also permute the blocks W_k within an optimal W (and correspondingly permute the sparse codes) to produce equivalent optimal solutions.

We note that in spite of sharing the common theme of a mixture of models, our OCTOBOS model and learning formulation are quite different from the Gaussian Mixture Model (GMM) approach of Zoran and Weiss [8], and Yu et al. [187]. In the GMM-based models, the signal can be thought of (cf. [8]) as approximated by a linear combination of a few (orthonormal) eigenvectors of the covariance matrix of the mixture component to which it belongs. In contrast, in the OCTOBOS approach, the transform blocks W_k (equivalently, the class-conditional square sparsifying transforms) are not eigenvectors of some covariance matrices. Instead they are directly optimized (via (P11.5)) for transform-domain sparsity of the training data. Our OCTOBOS learning also enforces well-conditioning rather than exact orthonormality of the transform blocks. These features distinguish our OCTOBOS framework from the GMM-based approach.

11.3 Transform Learning Algorithm and Properties

11.3.1 Algorithm

We propose an alternating algorithm to solve the joint minimization Problem (P11.5). In one step of our proposed algorithm called the *sparse coding and clustering step*, we solve for $\{C_k\}$, $\{X_i\}$ with fixed $\{W_k\}$ in (P11.5). In the other step of the algorithm called the *transform update step*, we solve for the transforms $\{W_k\}$ in (P11.5) with fixed sparse codes.

11.3.1.1 Sparse Coding and Clustering

Given the training matrix Y, and fixed transforms $\{W_k\}$ (or, the equivalent overcomplete W), we solve the following Problem (P11.6) (which is just (P11.5) with fixed transforms) to determine the sparse codes and clusters. As before, the clusters are disjoint and every training signal belongs to exactly one cluster.

$$\text{(P11.6)} \quad \min_{\{C_k\},\{X_i\}} \sum_{k=1}^{K} \sum_{i \in C_k} \left\{ \|W_k Y_i - X_i\|_2^2 + \eta_k \|Y_i\|_2^2 \right\}$$
$$s.t. \;\; \|X_i\|_0 \leq s \;\; \forall \; i, \;\; \{C_k\} \in G$$

The weight $\eta_k = \lambda_0 \, Q(W_k)$ above. This is a fixed weight, since W_k is fixed in this step. We refer to the term $\|W_k Y_i - X_i\|_2^2 + \eta_k \|Y_i\|_2^2$, with $X_i = H_s(W_k Y_i)$ (i.e., the optimal sparse code of Y_i in transform W_k) as a clustering measure corresponding to the signal Y_i. This is a modified version of the measure in (P11.1), and includes the additional penalty $\eta_k \|Y_i\|_2^2$ determined by the regularizer (i.e., determined by the conditioning of W_k [8]). It is easy to observe that the objective in (P11.6) involves the summation of only N such 'clustering measure' terms (one for each signal). Since every training signal is counted exactly once (in one cluster) in the double summation in Problem

[8]This clustering measure will encourage the shrinking of clusters corresponding to any badly conditioned, or badly scaled transforms.

(P11.6), we can construct the equivalent optimization problem as follows.

$$\sum_{i=1}^{N} \min_{1\leq k\leq K} \left\{ \|W_k Y_i - H_s(W_k Y_i)\|_2^2 + \eta_k \|Y_i\|_2^2 \right\} \tag{11.9}$$

The minimization over k for each Y_i above determines the cluster C_k (in (P11.6)) to which Y_i belongs. For each Y_i, the optimal cluster index $\hat{k}_i$ [9] is such that $\left\|W_{\hat{k}_i} Y_i - H_s(W_{\hat{k}_i} Y_i)\right\|_2^2 + \eta_{\hat{k}_i} \|Y_i\|_2^2 \leq \|W_j Y_i - H_s(W_j Y_i)\|_2^2 + \eta_j \|Y_i\|_2^2$, $\forall j \neq \hat{k}_i$. The optimal $\hat{X}_i$ in (P11.6) is then $H_s\left(W_{\hat{k}_i} Y_i\right)$. There is no coupling between the sparse coding/clustering problems in (11.9) for the different training signals $\{Y_i\}_{i=1}^{N}$. Thus, the training signals can be sparse coded and clustered, in parallel.

11.3.1.2 Transform Update Step

Here, we solve for $\{W_k\}$ in (P11.5) with fixed $\{C_k\}$, $\{X_i\}$. Although this is an unconstrained joint minimization problem over the set of transforms, the optimization problem is actually separable (due to the objective being in summation form) into K unconstrained problems, each involving only a particular square transform W_k. Thus, the transform update problem becomes

$$\text{(P11.7)} \quad \min_{W_k} \sum_{i\in C_k} \|W_k Y_i - X_i\|_2^2 + \lambda_k Q(W_k)$$

Here, $\lambda_k = \lambda_0 \|Y_{C_k}\|_F^2$ is a fixed weight. Problem (P11.7) is solved separately for each k, which can be done in parallel. Problem (P11.7) is similar to the transform update problem encountered for (P11.3) [6], and can thus be solved similarly.

Let U be the matrix whose columns are the training signals Y_i belonging to the k$^{\text{th}}$ cluster (i.e., $i \in C_k$). Let V be the corresponding (fixed) sparse code matrix. Problem (P11.7) can then be solved exactly and efficiently using simple closed-form solutions [6]. First, we decompose the positive-definite matrix $UU^T + \lambda_k I$ as $UU^T + \lambda_k I = LL^T$ (e.g., by Cholesky decomposition, or taking the eigenvalue decomposition (EVD) square root), where I is the

[9] When two or more clusters are equally optimal, then we pick the one corresponding to the lowest cluster index k.

$n \times n$ identity. Next, we obtain the full singular value decomposition (SVD) of the matrix $L^{-1}UV^T$ as $B\Sigma R^T$, where B, Σ, and R are all $n \times n$ matrices. Then, the optimal transform $\hat{W}_k$ in (P11.7) is given as

$$\hat{W}_k = \frac{R}{2}\left(\Sigma + (\Sigma^2 + 2\lambda_k I)^{\frac{1}{2}}\right) B^T L^{-1} \tag{11.10}$$

where the $(\cdot)^{\frac{1}{2}}$ operation above denotes the positive definite square root. The closed-form solution (11.10) is guaranteed to be a global optimum of Problem (P11.7). Compared to iterative optimization methods such as conjugate gradients (CG), (11.10) allows for both cheap and exact computation of the solution of the transform update problem. The OCTOBOS learning Algorithm A1 is summarized in Fig. 11.1. Algorithm A1 assumes that an initial estimate $\left\{\hat{W}_k^0, \hat{X}_i^0, \hat{C}_k^0\right\}$ is available (see Section 11.6.2 for examples of initializations). The initial $\left\{\hat{W}_k^0\right\}$ is only used by the algorithm in a degenerate scenario mentioned later in Footnote 16 of this chapter.

11.3.2 Computational Cost

The algorithm for (P11.5) consists of the sparse coding and clustering step, and the transform update step. We derive the computational cost of each of these steps.

Sparse coding and clustering. First, the sparse code of every training signal with respect to every transform W_k is computed. The computation of $W_k Y$ requires $n^2 N$ multiply-add operations. The projection of $W_k Y$ on to the s-ℓ_0 ball if done by full sorting would require $O(nN \log n)$ operations. Thus, the cost of finding the sparse representation of the training matrix in a particular W_k is dominated by $O(n^2 N)$. Since this needs to be done for each k, the total number of operations scales as $O(Kn^2 N)$. Denoting $m = Kn$ this cost is $O(mnN)$.

Next, for each training signal, in order to perform clustering, the clustering measure, which is the sum of the sparsification error and weighted signal energy, is computed with respect to all the transforms. The total cost of computing the sparsification error (taking into account that the sparsification error is only computed using the transform domain residual on the complement of the support of the sparse code) for all training signals in all clusters is

```
OCTOBOS Learning Algorithm A1
Input : $Y$ - training matrix with $N$ signals, $s$ - sparsity level, $\lambda_0$ - constant, $K$ - number of clusters, $J$ - number of iterations.
Output : $\{\hat{W}_k\}$ - learned transforms, $\{\hat{X}_i\}$ - learned sparse codes, $\{\hat{C}_k\}$ - learned clusters.
Initial Estimates: $\{\hat{W}_k^0, \hat{X}_i^0, \hat{C}_k^0\}$.
For $t = 1$ : J Repeat
1) Transform Update: For each $1 \le k \le K$, do
    (a) Let $\Psi \triangleq \hat{C}_k^{t-1}$. Compute $\lambda_k = \lambda_0 \|Y_\Psi\|_F^2$.
    (b) Compute $L^{-1} = \left(Y_\Psi Y_\Psi^T + \lambda_k I\right)^{-1/2}$.
    (c) Compute full SVD of $L^{-1} Y_\Psi (\hat{X}_\Psi^{t-1})^T$ as $B \Sigma R^T$.
    (d) $\hat{W}_k^t = \frac{R}{2}\left(\Sigma + (\Sigma^2 + 2\lambda_k I)^{\frac{1}{2}}\right) B^T L^{-1}$.
2) Sparse Coding and Clustering: For $1 \le i \le N$,
    (a) If $i = 1$, set $\hat{C}_k^t = \emptyset \ \forall \ k$, and compute $\eta_k = \lambda_0 \, Q(\hat{W}_k^t)$, $1 \le k \le K$.
    (b) Compute $\gamma_k = \eta_k \|Y_i\|_2^2 + \left\|\hat{W}_k^t Y_i - H_s(\hat{W}_k^t Y_i)\right\|_2^2$, $1 \le k \le K$. Set $\hat{k} = \min\{k : \gamma_k = \min_k \gamma_k\}$. Set $\hat{C}_{\hat{k}}^t \leftarrow \hat{C}_{\hat{k}}^t \cup \{i\}$.
    (c) $\hat{X}_i^t = H_s\left(\hat{W}_{\hat{k}}^t Y_i\right)$.
End
```

Figure 11.1: Algorithm A1 for OCTOBOS learning via (P11.5). A superscript of t is used to denote the iterates in the algorithm.

	Square Trans.	OCTOBOS	KSVD
Cost	$O(n^2N)$	$O(mnN)$	$O(mn^2N)$

Table 11.1: Computational cost per-iteration for square sparsifying transform, OCTOBOS, and KSVD learning.

$O((n-s)KN)$. Since the weighted signal energy term $\lambda_0 \|Y_i\|_2^2 \, Q(W_k)$ needs to be computed for all i, k, we first compute $\lambda_0 \, Q(W_k)$ for all k at a cost of $O(Kn^3)$ (computing determinants dominates this cost). Next, the term $\|Y_i\|_2^2$ is computed for all i at a cost of $O(nN)$. Then, computing $\lambda_0 \|Y_i\|_2^2 \, Q(W_k)$ and adding it to the corresponding sparsification error term for all i, k, requires $O(KN)$ operations. Finally, to compute the best cluster for each signal requires $O(K-1)$ comparisons (between the clustering measure values for different transforms), and thus $O((K-1)N)$ operations for the entire Y matrix. Assuming, $N \gg n$, it is clear based on the preceding arguments that the computation of $\{W_kY\}$ dominates the computations in the sparse coding and clustering step, with a cost of $O(mnN)$ (or, $O(Kn^2N)$).

Transform update. In this step, we compute the closed-form solution for the transform in each cluster using (11.10). First, the matrix $UU^T + \lambda_k I_n$ (notations defined in Section 11.3.1.2) needs to be computed for each (disjoint) cluster. This requires $O(n^2N)$ multiply-add operations totally (over all clusters). (Note that the computation of all the λ_k's requires only $O(nN)$ operations.) The computation of L and L^{-1} requires $O(n^3)$ operations for each cluster, and thus about $O(Kn^3)$ operations for K clusters. Next, the matrix UV^T is computed for each cluster. Since V has s-sparse columns, this matrix multiplication gives rise to a total (over all clusters) of $\alpha n^2 N$ multiply-add operations (assuming $s = \alpha n$, with $\alpha < 1$). Finally, the computation of $L^{-1}UV^T$, its SVD, and the closed-form update (11.10) require $O(n^3)$ operations per cluster, or about $O(Kn^3)$ operations for K clusters. Since, $N \gg m = Kn$ typically, we have that the cost of the transform update step scales as $O(n^2N)$. Thus, for $K \gg 1$, the transform update step is cheaper than the sparse coding step for the proposed algorithm.

Based on the preceding arguments, it is clear that the computational cost per iteration (of sparse coding and transform update) of our algorithm scales as $O(mnN)$ [10]. This is much lower (in order) than the per-iteration cost of

[10]Setting $m = n$ for the case $K = 1$, this agrees with previous cost analysis for square transform learning using (P11.3), which has per-iteration cost of $O(n^2N)$ [9].

learning an $n \times m$ overcomplete synthesis dictionary D using K-SVD [17], which, (assuming, as in the transform model, that the synthesis sparsity level $s = \beta n$ with $\beta < 1$ [11]), scales as $O(mn^2N)$. Our transform learning also holds a similar (per-iteration) computational advantage over analysis dictionary learning schemes such as analysis K-SVD. The computational costs per-iteration of square transform, OCTOBOS, and KSVD learning are summarized in Table 11.1.

As illustrated in our experiments in Section 11.6, both the OCTOBOS and square transform learning algorithms converge in few iterations in practice. Therefore, the per-iteration computational advantages for OCTOBOS over K-SVD typically translate to a net computational advantage in practice.

OCTOBOS learning could be used for a variety of purposes including clustering (classification), denoising, and sparsity-based signal compression. In the latter case, we also need to compute $\hat{Y}_i = W_k^{-1}X_i$, for all $i \in C_k$, and $\forall\ k$, in order to recover estimates of the signals from their (compressed) transform codes. Computing W_k^{-1}, $1 \leq k \leq K$, has $O(Kn^3)$ computational cost. However, this cost does not depend on the number of training signals $N \gg Kn$ (typically), and is therefore negligible compared to the total cost $O(snN)$ $(= O(n^2N)$ for $s \propto n)$ of multiplying the once computed W_k^{-1} for all k, with the corresponding sparse codes. The latter cost is the same as for multiplying a synthesis dictionary D with its sparse codes.

11.4 Convergence Analysis

In this section, we analyze the convergence behavior of the proposed OCTOBOS learning algorithm, that solves (P11.5). While some recent works [188, 189, 190, 191, 192, 193] study the convergence of (specific) synthesis dictionary learning algorithms [12], none of them consider the union of dictio-

[11]The notion that sparsity s scales with the signal dimension n is rather standard. For example, while $s = 1$ may work for representing the 4×4 patches of an image in a DCT dictionary with $n = 16$, the same sparsity level of $s = 1$ for an $n = 256^2$ DCT dictionary for a 256×256 (vectorized) image would lead to very poor image representation. Therefore, the sparsity s must increase with the size n. A typical assumption is that the sparsity s scales as a fraction (e.g., 5% or 10%) of the image or, patch size n. Otherwise, if s were to increase only sub-linearly with n, it would imply that larger (more complex) images are somehow better sparsifiable, which is not true in general.

[12]Most of these (synthesis dictionary learning) algorithms have not been demonstrated to be practically useful in applications such as denoising. Bao et al. [192] show that their

naries case. The prior works also typically require many restrictive assumptions (e.g., noiseless data) for their results to hold. In contrast, we present a novel convergence theory here for learning a union of (transform) sparse models. Importantly, although our proposed Problem formulation (P11.5) is highly non-convex (due to the ℓ_0 "norm" on the sparse codes and the log-determinant penalty), our results hold with few or no assumptions.

Very recently, a convergence result [96] has been derived for the algorithm for the (single) square transform learning Problem (P11.3). Here, we derive the results for the general overcomplete OCTOBOS case.

11.4.1 Main Results

The convergence results are more conveniently stated in terms of an unconstrained form of (P11.5). Problem (P11.5) has the constraint $\|X_i\|_0 \leq s \, \forall \, i$, which can equivalently be added as a penalty in the objective by using a barrier function $\phi(X)$ (where $X \in \mathbb{R}^{n \times N}$ is the matrix of all sparse codes), which takes the value $+\infty$ when the constraint is violated, and is zero otherwise. Then, we denote the objective of (P11.5) as

$$\begin{aligned} g\,(W, X, \Gamma) = & \sum_{k=1}^{K} \sum_{i \in C_k} \|W_k Y_i - X_i\|_2^2 + \phi(X) \\ & + \sum_{k=1}^{K} \lambda_k Q(W_k) \end{aligned} \tag{11.11}$$

where $W \in \mathbb{R}^{Kn \times n}$ is obtained by stacking the W_k's on top of one another, the matrix $X \in \mathbb{R}^{n \times N}$ contains the sparse codes X_i as its columns, and the row vector $\Gamma \in \mathbb{R}^{1 \times N}$ is such that its i$^{\text{th}}$ element $\Gamma_i \in \{1, .., K\}$ denotes the cluster index (label) corresponding to the signal Y_i. As discussed in Section 11.2.4, the clusters $\left\{C_k\right\}$ form a disjoint partitioning of $[1 : N]$.

Problem (P11.5) is to find the best possible union of sparsifying transforms model for a given set of data Y, by minimizing the sparsification error, and controlling the condition numbers (avoiding trivial solutions) of the cluster-specific transforms. Proposition 18 in Section 11.2.4 established the identifiability of good models by solving Problem (P11.5). Here, we discuss

method denoises worse than the K-SVD method [1].

the convergence behavior of our algorithm A1 that solves (P11.5).

For our convergence results, we make only the following mild assumption.

Assumption 1. Our proposed solution involves computing SVDs, EVDs (of small $n \times n$ matrices), and scalar square roots. Although in practice these quantities are computed using iterative methods, we assume, for the theoretical analysis, that they are computed exactly. In practice, standard numerical methods are guaranteed to quickly provide machine precision accuracy for these computations.

Since the training matrix $Y \in \mathbb{R}^{n\times N}$, the training signals are also bounded, i.e., $\max_i \|Y_i\|_2 < \infty$. For a vector z, let $\beta_j(z)$ denote the magnitude of the j^{th} largest element (magnitude-wise) of z. For a matrix B, $\|B\|_\infty \triangleq \max_{i,j} |B_{ij}|$. We say that a sequence $\{b^t\}$ has an accumulation point b, if there is a subsequence that converges to b. For our iterative Algorithm A1 (in Fig. 11.1), we denote the iterates (or, outputs) at each iteration t by the set (W^t, X^t, Γ^t), where W^t denotes the matrix obtained by stacking the cluster-specific transforms W_k^t $(1 \leq k \leq K)$, X^t is the sparse code matrix with the sparse codes X_i^t $(1 \leq i \leq N)$ as its columns, and Γ^t is a row vector containing the cluster indices Γ_i^t $(1 \leq i \leq N)$ as its elements. Each Γ_i^t contains the cluster index corresponding to signal Y_i. The following theorem provides a convergence result for the OCTOBOS learning Algorithm A1.

Theorem 8. *Let $\{W^t, X^t, \Gamma^t\}$ denote the iterate sequence generated by Algorithm A1 with training data Y and initial (W^0, X^0, Γ^0). Then, the objective sequence $\{g^t\}$ with $g^t \triangleq g(W^t, X^t, \Gamma^t)$ is monotone decreasing, and converges to a finite value, say $g^* = g^*(W^0, X^0, \Gamma^0)$. The iterate sequence is bounded, and all its accumulation points are equivalent in the sense that they achieve the same value g^* of the objective. Finally, every accumulation point (W, X, Γ) is a fixed point of Algorithm A1 satisfying the following partial global optimality conditions*

$$(X, \Gamma) \in \underset{\tilde{X}, \tilde{\Gamma}}{\arg\min}\; g\left(W, \tilde{X}, \tilde{\Gamma}\right) \tag{11.12}$$

$$W \in \underset{\tilde{W}}{\arg\min}\; g\left(\tilde{W}, X, \Gamma\right) \tag{11.13}$$

as well as the following partial local optimality property.

$$g(W + dW, X + \Delta X, \Gamma) \geq g(W, X, \Gamma) \tag{11.14}$$

Property (11.14) *holds for all* dW *with sufficiently small perturbations to the individual cluster-specific transforms* $dW_k \in \mathbb{R}^{n\times n}$ *satisfying* $\|dW_k\|_F \leq \epsilon_k$ *for some* $\epsilon_k > 0$ *that depends on the specific* W_k*, and the following condition on* ΔX*. For every* $1 \leq k \leq K$*, let* $\Delta X_{C_k} \in \mathbb{R}^{n\times|C_k|}$ *be the matrix with columns* $\Delta X_i \in \mathbb{R}^n$ *for* $i \in C_k$*. Then,* ΔX *is such that* $\Delta X_{C_k} \in \mathrm{R1_k} \cup \mathrm{R2_k}$ *for all* k*, where*

R1$_k$*: The half-space tr* $\left\{(W_k Y_{C_k} - X_{C_k})\Delta X_{C_k}^T\right\} \leq 0$.

R2$_k$*: The* local region *defined by*
$\|\Delta X_{C_k}\|_\infty < \min_{i\in C_k}\left\{\beta_s(W_k Y_i) : \|W_k Y_i\|_0 > s\right\}$.

Furthermore, if we have $\|W_k Y_i\|_0 \leq s\, \forall\, i \in C_k$*, then the corresponding* ΔX_{C_k} *can be arbitrary.*

The local region R2$_k$ in Theorem 8 that determines the size of the local perturbation ΔX_{C_k} in class k, is determined by the scalar $\gamma_k = \min_{i\in C_k}\left\{\beta_s(W_k Y_i) : \|W_k Y_i\|_0 > s\right\}$. This scalar is computed by (i) taking the columns of $W_k Y$ corresponding to the kth cluster; (ii) choosing only the vectors with sparsity $> s$; (iii) finding the sth largest magnitude element of those vectors; and (iv) picking the smallest of those values.

Theorem 8 indicates that for a particular starting point (W^0, X^0, Γ^0), the iterate sequence in our algorithm converges to an equivalence class of accumulation points. Every accumulation point has the same cost $g^* = g^*(W^0, X^0, \Gamma^0)$ [13], and is a fixed point of the algorithm, as well as a partial local minimizer (with respect to the cluster transforms and sparse code variables) of the objective $g(W, X, \Gamma)$. Since Algorithm A1 minimizes the objective $g(W, X, \Gamma)$ by alternating between the minimization over (X, Γ) and W, and obtains the global optimum in each of these minimizations, it follows that the algorithm's fixed points satisfy the partial global optimality conditions (11.12) and (11.13).

Thus, we can also say that, for each initial (W^0, X^0, Γ^0), the iterate sequence in OCTOBOS converges to an equivalence class of fixed points, or an equivalence class of partial local/global minimizers satisfying (11.12), (11.13), and (11.14). This is summarized by the following Corollary.

[13]The exact value of g^* may vary with initialization. We will empirically illustrate in Section 11.6.2 that our algorithm is also insensitive to initialization.

Corollary 9. *For a particular initial* (W^0, X^0, Γ^0)*, the iterate sequence in Algorithm A1 converges to an equivalence class of fixed points, that are also partial minimizers satisfying* (11.12)*,* (11.13)*, and* (11.14)*.*

The following corollary summarizes the convergence of Algorithm A1 for any starting point (the phrase "globally convergent" refers to convergence from any initialization).

Corollary 10. *The iterate sequence in Algorithm A1 is globally convergent to the set of partial minimizers of the non-convex objective* $g(W, X, \Gamma)$*.*

We would like to emphasize that unlike results in previous work (for synthesis learning), Theorem 8 that shows the convergence of the proposed non-convex OCTOBOS learning algorithm is free of any restrictive conditions or requirements. Theorem 8 also holds for any choice of the parameter λ_0 in (P11.5), which controls the condition number of the cluster transforms. The condition (11.14) in Theorem 8 holds true not only for local (or small) perturbations in the sparse codes, but also for arbitrarily large perturbations of the sparse codes in a half space, as defined by region $\mathrm{R1}_k$.

While Theorem 8 shows partial local/global optimality for Algorithm A1, the following Theorem 9 establishes that, under certain conditions, every accumulation point of the iterate sequence in Algorithm A1 is a stationary point of the overall objective. Algorithm A1 then converges to the set of stationary points of the overall problem.

Equation (11.9) indicates that the objective that is minimized in Problem (P11.5) can be equivalently written as

$$f(W) = \sum_{i=1}^{N} \min_{k} \left\{ \|W_k Y_i - H_s(W_k Y_i)\|_2^2 + \lambda_0 \, Q(W_k) \, \|Y_i\|_2^2 \right\} \tag{11.15}$$

This equivalent objective is now only a function of the transforms $\{W_k\}$ (with the cluster assignment being implicit). Our OCTOBOS learning algorithm can be thought of as an alternating minimization algorithm to minimize the function $f(W)$, that also involves the optimization with respect to the additionally introduced sparse code and cluster index variables.

We now state Theorem 9 in terms of the equivalent objective $f(W)$.

Theorem 9. *Let $\{W^t, X^t, \Gamma^t\}$ denote the iterate sequence generated by Algorithm A1 with training data Y and initial (W^0, X^0, Γ^0). Let each accumulation point (W, X, Γ) of the iterate sequence be such that (X, Γ) is the* unique *minimizer of $g\left(W, \tilde{X}, \tilde{\Gamma}\right)$ for fixed W. Then, every accumulation point of the iterate sequence is a stationary point of the objective $f(W)$.*

Theorem 9 establishes that the iterates converge to the set of stationary points of $f(W)$. It assumes that for every accumulation point (W, X, Γ), the pair (X, Γ) is the *unique* minimizer of $g\left(W, \tilde{X}, \tilde{\Gamma}\right)$ for fixed W. Note that the condition $(X, \Gamma) \in \underset{\tilde{X}, \tilde{\Gamma}}{\arg\min}\ g\left(W, \tilde{X}, \tilde{\Gamma}\right)$ is already guaranteed by Theorem 8. Only the uniqueness of the sparse coding and clustering solution is additionally assumed in Theorem 9, i.e., we assume that there are no ties in assigning the clusters or sparse codes.

Although the result in Theorem 9 depends on the uniqueness condition, the following conjecture postulates that provided the following Assumption 2 (that uses a probabilistic model for the data) holds, the uniqueness condition holds with probability 1, i.e., the probability of a tie in assigning the cluster or sparse code is zero.

Assumption 2. The training signals $Y_i \in \mathbb{R}^n$ for $1 \leq i \leq N$, are drawn independently from an absolutely continuous probability measure over the n-dimensional ball $\hat{S} \triangleq \{y \in \mathbb{R}^n : \|y\|_2 \leq c_0\}$ for some $c_0 > 0$.

Conjecture 1. *Let Assumption 2 hold. Then, with probability 1, every accumulation point (W, X, Γ) of Algorithm A1 is such that (X, Γ) is the* unique *minimizer of $g\left(W, \tilde{X}, \tilde{\Gamma}\right)$ for fixed W.*

Conjecture 1 is motivated in Section 11.4.2.3. If Conjecture 1 holds, then every accumulation point of the iterate sequence in Algorithm A1 is immediately a stationary point of $f(W)$ with probability 1.

11.4.2 Proofs

We use the operation $\tilde{H}_s(b)$ here to denote the *set* of all optimal projections of $b \in \mathbb{R}^n$ onto the s-ℓ_0 ball defined as $\{x \in \mathbb{R}^n : \|x\|_0 \leq s\}$.

We now prove the following properties of Algorithm A1 one-by-one.

(i) The objective sequence in Algorithm A1 converges.

(ii) The sequence of iterates is bounded.

(iii) The iterate sequence has an accumulation point.

(iv) All the accumulation points of the iterate sequence have a common objective value.

(v) Every accumulation point of the iterate sequence is a fixed point of the algorithm satisfying the partial global optimality conditions (11.12) and (11.13).

(vi) Every fixed point of the algorithm is a local minimizer of $g(W, X, \Gamma)$ with respect to the transforms $\{W_k\}$ and sparse codes $\{X_i\}$.

(vii) Under the uniqueness condition stated in Theorem 9, every accumulation point is a stationary point of the equivalent objective $f(W)$.

Items (i)-(vi) above pertain to Theorem 8 and establish the various results in Theorem 8, while item (vii) pertains to Theorem 9.

11.4.2.1 Proof of Theorem 8

Our first lemma shows the convergence of the objective sequence in Algorithm A1.

Lemma 7. *Let $\{W^t, X^t, \Gamma^t\}$ denote the iterate sequence generated by Algorithm A1 with training data Y and initial (W^0, X^0, Γ^0). Then, the objective sequence $\{g^t\}$ with $g^t \triangleq g(W^t, X^t, \Gamma^t)$ is monotone decreasing, and converges to a finite value, say $g^* = g^*(W^0, X^0, \Gamma^0)$.*

Proof. : In the transform update step, we solve K independent unconstrained problems. For each k, we obtain a global minimizer with respect to W_k in (P11.7). The closed-form solution is given in (11.10). Since, by Assumption 1, we obtain an exact solution in the transform update step, the objective decreases in this step, i.e., $g(W^{t+1}, X^t, \Gamma^t) \leq g(W^t, X^t, \Gamma^t)$. Furthermore, as discussed in Section 11.3.1.1, in the sparse coding and clustering step too, we obtain an exact solution with respect to $\{C_k\}$ and $\{X_i\}$ (for fixed cluster transforms). Therefore, $g\left(W^{t+1}, X^{t+1}, \Gamma^{t+1}\right) \leq g\left(W^{t+1}, X^{t}, \Gamma^{t}\right)$. Combining the results for the two steps, we get

$$g\left(W^{t+1}, X^{t+1}, \Gamma^{t+1}\right) \leq g\left(W^{t}, X^{t}, \Gamma^{t}\right) \tag{11.16}$$

By Lemma 6 in Section 11.2.4, the objective in (P11.5) is lower bounded. Since, the objective is monotone decreasing and lower bounded, it converges.

□

The next lemma establishes the boundedness of the iterate sequence.

Lemma 8. *The iterate sequence generated by Algorithm A1 is bounded.*

Proof. : For each t, the iterate is (W^t, X^t, Γ^t). Clearly, $1 \leq \Gamma_i^t \leq K$. Therefore, it is obvious that Γ_i^t is bounded by K for any i and all t.

Now, consider the triplet $(W^t, X^{t-1}, \Gamma^{t-1})$. Since $g\left(W^t, X^{t-1}, \Gamma^{t-1}\right)$ $= \sum_{k=1}^{K} \sum_{i \in C_k^{t-1}} \left\| W_k^t Y_i - X_i^{t-1} \right\|_2^2 + \sum_{k=1}^{K} \lambda_0 \left\| Y_{C_k^{t-1}} \right\|_F^2 Q(W_k^t)$ (note that $\phi(X) = 0$ for the iterates) is a sum of non-negative terms [14], we have that for any k,

$$\lambda_0 \left\| Y_{C_k^{t-1}} \right\|_F^2 Q(W_k^t) \leq g\left(W^t, X^{t-1}, \Gamma^{t-1}\right) \leq g^0 \tag{11.17}$$

where the last inequality is due to the monotonic decrease of the objective (Lemma 7). Now, it could happen that at a particular iteration, the cluster C_k^{t-1} (the output of the clustering step) is empty. In such cases, we assume that in the subsequent transform update step, the transform W_k^t for the k$^{\text{th}}$ cluster remains fixed at W_k^{t-1}. This transform is still used in the following clustering step, and may produce a non-empty cluster C_k^t. Since W_k^t remains fixed whenever C_k^{t-1} is empty, we only need to bound it for the iterations where C_k^{t-1} is non-empty.

Now, assuming C_k^{t-1} is non-empty, we can further consider two sub-cases: (1) $\left\| Y_{C_k^{t-1}} \right\|_F = 0$; and (2) $\left\| Y_{C_k^{t-1}} \right\|_F \neq 0$. Now, when $\left\| Y_{C_k^{t-1}} \right\|_F = 0$, it means that the signals in the k$^{\text{th}}$ cluster are all zero. In this case, the corresponding sparse codes $X_{C_k^{t-1}}^{t-1}$ (obtained by thresholding zero signals) will also be all zero. Therefore, the objective for the k$^{\text{th}}$ cluster in the t$^{\text{th}}$ transform update step is identically zero in this case. This objective is minimized by any (transform) matrix in $\mathbb{R}^{n \times n}$. Therefore, in this case, we assume that the

[14]The regularizer $Q(W_k^t)$ is non-negative by the arguments in the proof of Lemma 6 in Section 11.2.4.

optimal W_k^t is set to W_k^{t-1}. Since W_k^t remains fixed in this case, we only need to consider the case when $\left\|Y_{C_k^{t-1}}\right\|_F \neq 0$. In the latter case, we have by (11.17) that

$$Q(W_k^t) \leq \frac{g^0}{\lambda_0 \left\|Y_{C_k^{t-1}}\right\|_F^2} \leq \frac{g^0}{\lambda_0 \eta^2} \tag{11.18}$$

where the last inequality follows from $\left\|Y_{C_k^t}\right\|_F \geq \eta$, with $\eta \triangleq \min_{i:\, \|Y_i\|_2 \neq 0} \|Y_i\|_2$ being a fixed strictly positive number.

The function $Q(W_k^t) = \sum_{i=1}^n (\alpha_i^2 - \log\, \alpha_i)$, where α_i $(1 \leq i \leq n)$ are the (all positive) singular values of W_k^t, is a coercive function of the singular values, and therefore it has bounded lower level sets [15]. Combining this fact with (11.18), we get that there is a constant c (that depends only on g^0, λ_0, η) such that $\|W_k^t\|_F = \sqrt{\sum_{i=1}^n \alpha_i^2} \leq c$. Thus, the sequence $\{W^t\}$ of cluster transforms is bounded.

We now bound the sparse codes X_i^t $(1 \leq i \leq N)$ for all t. First, for each iteration t and index i, we have that there exists a k $(1 \leq k \leq K)$ such that $X_i^t = H_s\left(W_k^t Y_i\right)$ (see Fig. 11.1). Therefore, by the definition of $H_s(\cdot)$, we have

$$\left\|X_i^t\right\|_2 = \left\|H_s\left(W_k^t Y_i\right)\right\|_2 \leq \left\|W_k^t Y_i\right\|_2 \leq \left\|W_k^t\right\|_2 \left\|Y_i\right\|_2 \tag{11.19}$$

Since W_k^t (by aforementioned arguments) and Y_i are both bounded (by constants that do not depend on t), (11.19) implies that the sequence $\{X^t\}$ of sparse codes is also bounded. $\square$

Proposition 19. *The iterate sequence in Algorithm A1 has at least one convergent subsequence, or in other words, it has at least one accumulation point.*

Proof. : Since the iterate sequence is bounded, the existence of a convergent subsequence (for a bounded sequence) is a standard result. $\square$

The following property of the accumulation points of the iterates in our algorithm will be used to prove that all accumulation points are equivalent.

Lemma 9. *Any accumulation point (W^*, X^*, Γ^*) of the iterate sequence gen-*

[15]The lower level sets of a function $\hat{f} : A \subset \mathbb{R}^n \mapsto \mathbb{R}$ (where A is unbounded) are bounded if $\lim_{t\to\infty} \hat{f}(x^t) = +\infty$ whenever $\{x^t\} \subset A$ and $\lim_{t\to\infty} \|x^t\| = \infty$.

erated by Algorithm A1 satisfies

$$X_i^* \in \tilde{H}_s\left(W_{\Gamma_i^*}^* Y_i\right) \;\; \forall\, i \tag{11.20}$$

Proof. : Consider the subsequence $\{W^{q_t}, X^{q_t}, \Gamma^{q_t}\}$ (indexed by q_t), that converges to the accumulation point (W^*, X^*, Γ^*). We then have for each i $(1 \leq i \leq N)$

$$X_i^* = \lim_{t\to\infty} X_i^{q_t} = \lim_{t\to\infty} H_s\left(W_{\Gamma_i^{q_t}}^{q_t} Y_i\right) \tag{11.21}$$

where $\Gamma_i^{q_t}$ is the cluster index for Y_i at iteration q_t. Now, for each i, since the integer sequence $\left\{\Gamma_i^{q_t}\right\}$ converges to Γ_i^* (which is also an integer in $\{1, ..., K\}$), this implies that this convergence takes place in a finite number of iterations, i.e., there exists a $t_0 \in \mathbb{N}$ such that $\forall\, t \geq t_0$, we have $\Gamma_i^{q_t} = \Gamma_i^*$. Using this result in (11.21) yields

$$X_i^* = \lim_{t\to\infty} H_s\left(W_{\Gamma_i^*}^{q_t} Y_i\right) \in \tilde{H}_s\left(W_{\Gamma_i^*}^* Y_i\right) \tag{11.22}$$

where the containment on the right hand side of (11.22) follows from Lemma 54 of Appendix G.1. Indeed, since the vector sequence $\left\{W_{\Gamma_i^*}^{q_t} Y_i\right\}$ converges to $W_{\Gamma_i^*}^* Y_i$, by Lemma 54 the accumulation point of the sequence $\left\{H_s\left(W_{\Gamma_i^*}^{q_t} Y_i\right)\right\}$ above must lie in the (possibly non-singleton) set $\tilde{H}_s\left(W_{\Gamma_i^*}^* Y_i\right)$. $\square$

For the following lemma, we define $\hat{g}(W, X, \Gamma) \triangleq g(W, X, \Gamma) - \phi(X)$.

Lemma 10. *All the accumulation points of the iterate sequence generated by Algorithm A1 with a given initialization correspond to a common objective value g^*. Thus, they are equivalent in that sense.*

Proof. : Consider the subsequence $\{W^{q_t}, X^{q_t}, \Gamma^{q_t}\}$ (indexed by q_t), that converges to the accumulation point (W^*, X^*, Γ^*).

First, for each k, the matrix W_k^* is non-singular. This follows from (11.18) which implies that

$$-\log\left|\det W_k^t\right| \leq \frac{g^0}{\lambda_0 \eta^2} \tag{11.23}$$

and (11.23) further implies that $\left|\det W_k^t\right|$ is bounded away from zero for all t. Hence, due to the continuity of the determinant function, W_k^* (the limit point of a subsequence) is non-singular. We then have that

$$\begin{aligned}\lim_{t\to\infty} \hat{g}\left(W^{q_t}, X^{q_t}, \Gamma^{q_t}\right) &= \lim_{t\to\infty} \hat{g}\left(W^{q_t}, X^{q_t}, \Gamma^{*}\right) \\ &= \hat{g}\left(W^{*}, X^{*}, \Gamma^{*}\right) \qquad (11.24)\end{aligned}$$

where for the first equality in (11.24), we used the previously mentioned fact that, for each i, the convergence of the integer sequence $\left\{\Gamma_i^{q_t}\right\}$ implies that $\Gamma_i^{q_t} = \Gamma_i^{*}$ for sufficiently large t. The second equality in (11.24) follows because, for fixed cluster indices, the function $\hat{g}$, is continuous with respect to the cluster transforms and sparse code vectors at (W^*, X^*). The continuity is obvious from the fact that the sparsification error term in $\hat{g}$ is a quadratic, and the regularizer term in $\hat{g}$ is continuous at non-singular matrices W_k^*.

Next, by Lemma 9, the accumulation point X_i^* (the ith column of X^*) is s-sparse for all i. Furthermore, Algorithm A1 guarantees that $X_i^{q_t}$ is s-sparse for each t and all i. Therefore, the barrier function $\phi(X)$ is zero for the sparse code subsequence and for its accumulation point. It follows that (11.24) is equivalent to

$$\lim_{t\to\infty} g\left(W^{q_t}, X^{q_t}, \Gamma^{q_t}\right) = g\left(W^{*}, X^{*}, \Gamma^{*}\right) \qquad (11.25)$$

because g and $\hat{g}$ coincide in these equations. Finally, by Lemma 7, the left hand side limit above is in fact g^*. Therefore, we have that for any accumulation point (W^*, X^*, Γ^*) of the iterate sequence,

$$g\left(W^{*}, X^{*}, \Gamma^{*}\right) = g^{*} \qquad (11.26)$$

□

For a particular accumulation point (W^*, X^*, Γ^*) of the iterate sequence in our algorithm, the following result shows that the cluster index Γ_i^* is the optimal cluster index for signal Y_i with respect to the set of transforms $\{W_k^*\}$.

Lemma 11. *Any accumulation point* (W^*, X^*, Γ^*) *of the iterate sequence generated by Algorithm A1 satisfies for each* $1 \leq i \leq N$ *the inequality*

$$\begin{aligned}&\left\|W_{\Gamma_i^*}^{*} Y_i - H_s\left(W_{\Gamma_i^*}^{*} Y_i\right)\right\|_2^2 + \lambda_0 \left\|Y_i\right\|_2^2 Q\left(W_{\Gamma_i^*}^{*}\right) \leq \qquad (11.27)\\ &\lambda_0 \left\|Y_i\right\|_2^2 Q\left(W_j^{*}\right) + \left\|W_j^{*} Y_i - H_s\left(W_j^{*} Y_i\right)\right\|_2^2 \;\; \forall\, j \neq \Gamma_i^{*}.\end{aligned}$$

Proof. : Consider the subsequence $\{W^{q_t}, X^{q_t}, \Gamma^{q_t}\}$ (indexed by q_t), that converges to the accumulation point (W^*, X^*, Γ^*). Let us pick a specific index i. The convergence of the integer sequence $\left\{\Gamma_i^{q_t}\right\}$ then implies that $\Gamma_i^{q_t} = \Gamma_i^*$ for sufficiently large $t \geq t_0$. For each $t \geq t_0$, the sparse coding and clustering step of Algorithm A1 guarantees that $\Gamma_i^{q_t} = \Gamma_i^*$ is the optimal cluster index for signal Y_i, i.e.,

$$\begin{aligned}&\left\|W_{\Gamma_i^*}^{q_t} Y_i - X_i^{q_t}\right\|_2^2 + \lambda_0 \left\|Y_i\right\|_2^2 Q\left(W_{\Gamma_i^*}^{q_t}\right) \leq \\ &\lambda_0 \left\|Y_i\right\|_2^2 Q\left(W_j^{q_t}\right) + \left\|W_j^{q_t} Y_i - H_s\left(W_j^{q_t} Y_i\right)\right\|_2^2 \quad \forall\, j \neq \Gamma_i^*.\end{aligned} \tag{11.28}$$

We would like to take the limit $t \to \infty$ on both sides of the above inequality. Notice that the sequence $\left\{W_j^{q_t}\right\}$ converges to W_j^* for every j, and by Lemma 9, the limit point of the sequence $\left\{X_i^{q_t}\right\}$ satisfies $X_i^* \in \tilde{H}_s\left(W_{\Gamma_i^*}^* Y_i\right)$. This implies that

$$\begin{aligned}\lim_{t\to\infty} \left\|W_{\Gamma_i^*}^{q_t} Y_i - X_i^{q_t}\right\|_2^2 &= \left\|W_{\Gamma_i^*}^* Y_i - X_i^*\right\|_2^2 \\ &= \left\|W_{\Gamma_i^*}^* Y_i - H_s\left(W_{\Gamma_i^*}^* Y_i\right)\right\|_2^2\end{aligned} \tag{11.29}$$

where the last equality in (11.29) follows because every sparse code in the set $\tilde{H}_s\left(W_{\Gamma_i^*}^* Y_i\right)$ provides the same sparsification error, or in other words, X_i^* and $H_s\left(W_{\Gamma_i^*}^* Y_i\right)$ provide the same sparsification error in (11.29). Furthermore, for a fixed j, since, by Lemma 54 of Appendix G.1, every accumulation point of the sequence $\left\{H_s\left(W_j^{q_t} Y_i\right)\right\}$ lies in the set $\tilde{H}_s\left(W_j^* Y_i\right)$, we also easily have that

$$\lim_{t\to\infty} \left\|W_j^{q_t} Y_i - H_s\left(W_j^{q_t} Y_i\right)\right\|_2^2 = \left\|W_j^* Y_i - H_s\left(W_j^* Y_i\right)\right\|_2^2$$

Now, since W_j^* is non-singular, the regularizer term $Q\left(W_j^{q_t}\right)$ converges to $Q\left(W_j^*\right)$ for any j.

Thus, taking the limit $t \to \infty$ on both sides of (11.28), and using the above limits for each of the terms in (11.28), we immediately get the result (11.27) of the lemma. □

The following property of the accumulation points in our algorithm will be used to prove that every accumulation point is a fixed point. In the following lemma, C_k^* denotes the set of indices belonging to the kth cluster, for an accumulation point (W^*, X^*, Γ^*) of the iterate sequence in Algorithm A1.

Lemma 12. *Any accumulation point* (W^*, X^*, Γ^*) *of the iterate sequence generated by Algorithm A1 satisfies*

$$W^* \in \underset{W}{\arg\min}\ g\left(W, X^*, \Gamma^*\right) \tag{11.30}$$

Specifically, for each $1 \leq k \leq K$*, we have*

$$W_k^* \in \underset{W_k}{\arg\min} \sum_{i \in C_k^*} \|W_k Y_i - X_i^*\|_2^2 + \lambda_0 \left\|Y_{C_k^*}\right\|_F^2 Q(W_k) \tag{11.31}$$

Proof. : Consider the subsequence $\{W^{q_t}, X^{q_t}, \Gamma^{q_t}\}$ (indexed by q_t), that converges to the accumulation point (W^*, X^*, Γ^*). Assume without loss of generality that the sequence $\{W^{q_t+1}\}$ converges to say W^{**}. Otherwise we can work with a convergent subsequence (exists since the sequence is bounded) $\{W^{q_{r_t}+1}\}$, and the following proof technique still holds by considering the subsequence $\{W^{q_{r_t}}, X^{q_{r_t}}, \Gamma^{q_{r_t}}\}$. Note that the subsequence $\{W^{q_{r_t}}, X^{q_{r_t}}, \Gamma^{q_{r_t}}\}$ converges to the same limit (W^*, X^*, Γ^*) as the original $\{W^{q_t}, X^{q_t}, \Gamma^{q_t}\}$ above.

For (11.31) to hold, we need only consider k for which C_k^* is non-empty and $\left\|Y_{C_k^*}\right\|_F \neq 0$. For any other k, (11.31) is trivially true. Let us now pick a specific such k. Since the integer vector sequence $\{\Gamma^{q_t}\}$ converges, it follows that, there exists a $t_0 \in \mathbb{N}$ such that $\Gamma^{q_t} = \Gamma^*$ for all $t \geq t_0$. Hence,

$$C_k^{q_t} = C_k^* \ \ \forall\, t \geq t_0, \ \ \forall\, k \tag{11.32}$$

In the remainder of this proof, we consider only $t \geq t_0$. Because of (11.32), the data in the kth cluster does not change over the subsequence iterations q_t for $t \geq t_0$. Hence, in the transform update step (of Algorithm A1) at iteration $q_t + 1$ for $t \geq t_0$, the computed (unique) inverse EVD square root matrix has the form

$$L^{-1} = \left(Y_{C_k^*} Y_{C_k^*}^T + \lambda_0 \left\|Y_{C_k^*}\right\|_F^2 I\right)^{-1/2} \tag{11.33}$$

where the subscript C_k^* is used to denote the matrix whose columns are the signals corresponding to the indices in C_k^*.

Let $B_k^{q_t}\Sigma_k^{q_t}\left(R_k^{q_t}\right)^T$ denote the full SVD of $L^{-1}Y_{C_k^*}\left(X_{C_k^*}^{q_t}\right)^T$. The transform $W_k^{q_t+1}$ is computed in (11.10) in terms of the above SVD by the expression

$$W_k^{q_t+1} = \frac{R_k^{q_t}}{2}\left(\Sigma_k^{q_t} + \left(\left(\Sigma_k^{q_t}\right)^2 + 2\lambda_k I\right)^{\frac{1}{2}}\right)\left(B_k^{q_t}\right)^T L^{-1}$$

where $\lambda_k = \lambda_0 \left\|Y_{C_k^*}\right\|_F^2$.

For L^{-1} defined as in (11.33), and $W_k^{q_t+1}$ defined as above, and recalling the assumption that $\left\{W_k^{q_t+1}\right\}$ converges to W_k^{**} and $\left\{W_k^{q_t}, X_{C_k^*}^{q_t}\right\}$ converges to $\left(W_k^*, X_{C_k^*}^*\right)$ for the chosen subsequence, we have that all conditions for Lemma 55 in Appendix G.1 are satisfied. Therefore, we have

$$W_k^{**} \in \underset{W_k}{\arg\min} \sum_{i\in C_k^*} \left\|W_k Y_i - X_i^*\right\|_2^2 + \lambda_k Q(W_k) \tag{11.34}$$

The above result holds for every k. Therefore, since the objective $g\left(W, X^*, \Gamma^*\right)$ is the sum of the objectives corresponding to each cluster-wise transform, we have

$$W^{**} \in \underset{W}{\arg\min}\ g\left(W, X^*, \Gamma^*\right) \tag{11.35}$$

Now, by Lemma 10, we have that $g\left(W^*, X^*, \Gamma^*\right) = g^*$. Furthermore, applying the same arguments as previously used in (11.24), (11.25), and (11.26) to the (convergent) sequence $\left\{W^{q_t+1}, X^{q_t}, \Gamma^{q_t}\right\}$ ($\Gamma^{q_t} = \Gamma^*$ for $t \geq t_0$), we also get that $g\left(W^{**}, X^*, \Gamma^*\right) = g^*$. Since W^* achieves the same value of the objective (with fixed sparse codes and cluster indices) as W^{**}, and using (11.35), we immediately have that (11.30) holds. Equation (11.31) then trivially holds due to the separability of the objective in (11.30). $\square$

Lemma 13. *Every accumulation point of the iterate sequence generated by Algorithm A1 is a fixed point of the algorithm.*

Proof. : Consider the subsequence $\left\{W^{q_t}, X^{q_t}, \Gamma^{q_t}\right\}$ (indexed by q_t), that converges to the accumulation point $\left(W^*, X^*, \Gamma^*\right)$. We then have by Lemma 11 and Lemma 9 that

$$\left(X^*, \Gamma^*\right) \in \underset{X,\Gamma}{\arg\min}\ g\left(W^*, X, \Gamma\right) \tag{11.36}$$

To see this, note that Lemma 11 provides the optimality of the cluster index

Γ_i^* for signal Y_i for each i, for given W^*. Now, for a given (optimal) cluster index Γ_i^*, the set of optimal sparse codes for signal Y_i is given by $\tilde{H}_s\left(W^*_{\Gamma_i^*} Y_i\right)$. Since, by Lemma 9, $X_i^* \in \tilde{H}_s\left(W^*_{\Gamma_i^*} Y_i\right)$, we obtain (11.36).

Next, we have the result of Lemma 12 that

$$W^* \in \underset{W}{\arg\min}\ g\left(W, X^*, \Gamma^*\right) \tag{11.37}$$

In order to deal with any non-uniqueness of solutions in (11.37), we assume for Algorithm A1 that if a certain iterate W^{t+1} (fixed t) satisfies $g\left(W^{t+1}, X^t, \Gamma^t\right) = g\left(W^t, X^t, \Gamma^t\right)$, then we equivalently set $W^{t+1} = W^t$. Similarly, in order to deal with any non-uniqueness of solutions in (11.36), we assume that if $W^{t+1} = W^t$ holds (fixed t) and $g\left(W^{t+1}, X^{t+1}, \Gamma^{t+1}\right) = g\left(W^t, X^t, \Gamma^t\right)$, then we equivalently set $\Gamma^{t+1} = \Gamma^t$ and $X^{t+1} = X^t$ [16].

These assumptions and equations (11.37) and (11.36) imply that if we provide the accumulation point $\left(W^*, X^*, \Gamma^*\right)$ into Algorithm A1 as the initial iterate, the algorithm stays at the point. Therefore, the accumulation point is a fixed point. □

The fixed point property implies that every accumulation point $\left(W^*, X^*, \Gamma^*\right)$ is a global optimum of $g\left(W, X, \Gamma\right)$ with respect to either $\left\{W_k\right\}$, or $\left(X, \Gamma\right)$, with the other variables fixed. The following lemma establishes the local optimality (jointly with respect to the transform and sparse code variables) of the accumulation points.

Lemma 14. *Every fixed point $\left(W, X, \Gamma\right)$ of Algorithm A1 is a local optimum of the objective $g\left(W, X, \Gamma\right)$ with respect to $\left(W, X\right)$.*

Proof. : Since $\left(W, X, \Gamma\right)$ is a fixed point of Algorithm A1, we have that

$$W \in \underset{\tilde{W}}{\arg\min}\ g\left(\tilde{W}, X, \Gamma\right) \tag{11.38}$$

The above optimization problem (over the cluster transforms) involves an unconstrained objective, that is separable into the component-wise objectives

[16] This rule is trivially satisfied due to the way Algorithm A1 is written, except perhaps for the case when the superscript $t = 0$. In the latter case, if the rule is applicable, it means that the algorithm has already reached a fixed point (the initial $\left(W^0, X^0, \Gamma^0\right)$ is a fixed point), and therefore, no more iterations are performed. All aforementioned convergence results hold true for this degenerate case.

corresponding to each $\tilde{W}_k$, i.e., we can write

$$g\left(\tilde{W}, X, \Gamma\right) = \sum_{k=1}^{K} g_k\left(\tilde{W}_k, X_{C_k}\right) \tag{11.39}$$

where we denoted the set of indices i for which $\Gamma_i = k$ by C_k. Specifically, the component-wise objective is given as

$$\begin{aligned} g_k\left(\tilde{W}_k, X_{C_k}\right) = & \left\|\tilde{W}_k Y_{C_k} - X_{C_k}\right\|_F^2 + \lambda_k \left\|\tilde{W}_k\right\|_F^2 \\ & - \lambda_k \log \left|\det \tilde{W}_k\right| + \phi(X_{C_k}) \end{aligned} \tag{11.40}$$

where $\lambda_k = \lambda_0 \left\|Y_{C_k}\right\|_F^2$.

Each W_k in (11.38) is a global minimizer of the corresponding component-wise objective. Therefore, it provides a gradient value of 0 (necessary condition) for that objective. Thus, we have that

$$2W_k Y_{C_k} Y_{C_k}^T - 2X_{C_k} Y_{C_k}^T + 2\lambda_k W_k - \lambda_k W_k^{-T} = 0 \tag{11.41}$$

Since (W, X, Γ) is a fixed point of Algorithm A1, we also have that

$$(X, \Gamma) \in \underset{\tilde{X}, \tilde{\Gamma}}{\arg\min}\ g\left(W, \tilde{X}, \tilde{\Gamma}\right) \tag{11.42}$$

Therefore, we have for any k that

$$X_i \in \tilde{H}_s(W_k Y_i)\ \ \forall\ i \in C_k \tag{11.43}$$

Now, keeping the cluster indices Γ fixed, and using the definition of g_k in (11.40) along with (11.41) and (11.43), and applying Lemma 56 in Appendix G.1, we get that the following condition holds for each component-wise objective g_k.

$$g_k(W_k + dW_k, X_{C_k} + \Delta X_{C_k}) \geq g_k\left(W_k, X_{C_k}\right) \tag{11.44}$$

The condition holds for all sufficiently small $dW_k \in \mathbb{R}^{n\times n}$ satisfying $\|dW_k\|_F \leq \epsilon_k$ for some $\epsilon_k > 0$ that depends on the specific W_k, and $\Delta X_{C_k} \in \mathbb{R}^{n\times|C_k|}$ in the union of the following regions.

R1$_k$. The half-space $tr\left\{(W_k Y_{C_k} - X_{C_k})\Delta X_{C_k}^T\right\} \leq 0$.

R2$_k$. The *local region* defined by
$\|\Delta X_{C_k}\|_\infty < \min_{i \in C_k} \{\beta_s(W_k Y_i) : \|W_k Y_i\|_0 > s\}$.

If we have $\|W_k Y_i\|_0 \leq s \, \forall \, i \in C_k$, then ΔX_{C_k} can be arbitrary.

Finally, summing the result in (11.44) over all k, we get that

$$g\,(W + dW, X + \Delta X, \Gamma) \geq g\,(W, X, \Gamma) \tag{11.45}$$

The above condition (11.45) holds for $\|dW_k\|_F \leq \epsilon_k$ and ΔX_{C_k} in R1$_k \cup$ R2$_k$ for all k. Thus, the fixed point (W, X, Γ) is a local minimum of the objective $g\,(W, X, \Gamma)$, with respect to the cluster transform and sparse code variables. □

This concludes the proof of Theorem 8.

11.4.2.2 Proof of Theorem 9

Proof. : Let (W, X, Γ) be an accumulation point of the iterate sequence in Algorithm A1. The function $f(W)$ involves the summation of N terms of the form

$$f_i(W) = \min_k \left\{ \|W_k Y_i - H_s(W_k Y_i)\|_2^2 + \lambda_0 \, Q(W_k) \, \|Y_i\|_2^2 \right\}$$

The function $f_i(W)$ computes the minimum value of the clustering measure for Y_i with respect to the set of transforms $\{W_k\}$. Recall that W is obtained the stacking the W_k's on top of each other. Let us denote the clustering measure corresponding to signal Y_i in a fixed transform $B \in \mathbb{R}^{n \times n}$ by the function $\tilde{f}_i(B) = \|BY_i - H_s(BY_i)\|_2^2 + \lambda_0 \, Q(B) \, \|Y_i\|_2^2$. Then,

$$f_i(W) = \min_k \, \tilde{f}_i(W_k) \tag{11.46}$$

Lemma 8 established the boundedness of the iterate sequence in Algorithm A1. This implies that the accumulation (W, X, Γ) is also bounded (same bound works as for the iterates). By Lemma 13, the accumulation point is a fixed point. Since $Y \in \mathbb{R}^{n \times N}$, the training signals are all bounded. We can now use Lemma 57 of Appendix G.1 (which shows Lipschitz continuity, and therefore continuity of the sparsification error function), and the fact that

$Q(B)$ is continuous at full rank matrices B, to conclude that the function $\tilde{f}_i(B)$ is continuous at the point W_k for any k.

Now, the optimal cluster index for a signal Y_i with respect to the set of transforms (fixed point) $\{W_k\}$ is (assumed) unique, i.e., there is a non-zero separation between the smallest value of the clustering measure $\tilde{f}_i(W_k)$ (minimum over k) and the second smallest value. For signal Y_i, by the fixed point equation (11.42), the optimal cluster index with respect to the fixed point $\{W_k\}$ is $k = \Gamma_i$. If we perturb $\{W_k\}$ by sufficiently small $\{dW_k\}$, the minimum cluster index for Y_i with respect to $\{W_k + dW_k\}$ remains at (the unique) Γ_i due to the continuity of the function $\tilde{f}_i(B)$ at $B = W_k \ \forall\ k$ in (11.46), and because $\arg\min_k \tilde{f}_i(W_k)$ is unique ($= \Gamma_i$).

Therefore, for a particular Y_i, we have that $f_i(W) = \tilde{f}_i(W_{\Gamma_i})$ and $f_i(W + dW) = \tilde{f}_i(W_{\Gamma_i} + dW_{\Gamma_i})$ for all sufficiently small $\{dW_k\}$ (dW is formed by stacking the dW_k on top of each other). Further, since X is the (assumed) unique minimizer of $g\left(W, \tilde{X}, \Gamma\right)$ for fixed W and Γ, we also have that $\tilde{H}_s(W_{\Gamma_i} Y_i)$ is the singleton $H_s(W_{\Gamma_i} Y_i)$. It is then easy to obtain (using the definition of the derivative as a limit) the following two equations

$$\begin{aligned}\nabla_{W_{\Gamma_i}} f_i(W) =& 2W_{\Gamma_i} Y_i Y_i^T - 2H_s(W_{\Gamma_i} Y_i) Y_i^T \\ &+ \lambda_0 \|Y_i\|_2^2 \left(2W_{\Gamma_i} - W_{\Gamma_i}^{-T}\right)\end{aligned} \tag{11.47}$$

$$\nabla_{W_k} f_i(W) = 0 \ \ \forall\ k \neq \Gamma_i. \tag{11.48}$$

Therefore, for a particular index i, the derivative of $f_i(W)$ at the fixed point W exists and is given by the above expressions.

We can now show that the fixed point (W, X, Γ) of Algorithm A1 is a stationary point of $f(W)$. First, by the assumption that X is the unique minimizer of $g\left(W, \tilde{X}, \Gamma\right)$ for fixed W and Γ, we have that $H_s(W_k Y_i) = \tilde{H}_s(W_k Y_i)$ $\forall\ i \in C_k$ in (11.43), or that $X_i = H_s(W_k Y_i)\ \forall\ i \in C_k$. Then, using (11.47) and (11.48), we have that for any k for which C_k is non-empty, the derivative $\nabla_{W_k} f(W)$ at the fixed point (W, X, Γ) is $2W_k Y_{C_k} Y_{C_k}^T - 2X_{C_k} Y_{C_k}^T + 2\lambda_k W_k - \lambda_k W_k^{-T}$, where $\lambda_k = \lambda_0 \left\|Y_{C_k}\right\|_F^2$. When C_k is empty, the aforementioned derivative is 0.

By the fixed point equation (11.41), we then have that $\nabla_{W_k} f(W) = 0$ at each k, i.e., the fixed point is a stationary point of f. This concludes the proof of Theorem 9. $\square$

11.4.2.3 Motivation for Conjecture 1

Conjecture 1 says that every accumulation point (W, X, Γ) of the iterate sequence generated by Algorithm A1 is such that (X, Γ) is the *unique* minimizer of $g\left(W, \tilde{X}, \tilde{\Gamma}\right)$ for fixed W. By Lemma 13, we know that every accumulation point (W, X, Γ) is a fixed point of Algorithm A1.

Firstly, Conjecture 1 says that the clustering indices Γ computed with respect to the set $\{W_k\}$ are uniquely optimal, i.e., for each Y_i, there is a non-zero separation between the smallest value of the clustering measure $\tilde{f}_i(W_k)$ (minimum over k) and the second smallest value [17]. From the fixed point equations (11.41) and (11.43), we can see that each transform W_k is essentially a (non-linear) function of the signals in cluster k. Since the clusters are disjoint, and the signals in each cluster are independent and continuously distributed (by Assumption 2), we conjecture that the event that any two transforms W_k and W_j (for $k \neq j$) achieve the exact same (minimum) value of the clustering measure for a signal Y_i has probability 0.

Secondly, Conjecture 1 says that the set $\tilde{H}_s(W_{\Gamma_i} Y_i)$ of optimal sparse codes for signal Y_i is a singleton $\forall\ i$. In order for this to fail, the vector $W_{\Gamma_i} Y_i$ must have two entries of identical magnitude. However, because the full rank W_{Γ_i} is a function of the training signals in class k, and since the training signals are continuously distributed with an absolutely continuous probability measure (by Assumption 2), we conjecture that the event that two entries of $W_{\Gamma_i} Y_i$ have identical magnitude has probability 0.

11.5 Image Denoising

There are numerous applications that benefit from a good sparse model. Image denoising is an important and classical application that has been widely studied. The goal of denoising is to recover an estimate of an image $x \in \mathbb{R}^P$ (2D image represented as a vector) from its corrupted measurement $y = x + h$, where $h \in \mathbb{R}^P$ is a noise vector. Here, we consider h whose entries are i.i.d. Gaussian with zero mean and variance σ^2. We propose an adaptive image denoising framework in this section that exploits the proposed union of transforms model, or OCTOBOS model.

[17] The uniqueness of the cluster index for each signal Y_i in the iterations of Algorithm A1 for various data sets was empirically observed.

11.5.1 Problem Formulation

Similar to previous work on dictionary-based image denoising [1], we work with image patches. We model them as sparse in a transform domain. We allow overlapping patches, which provide an additional averaging effect that reduces noise. The patches considered can be vectorized to form the columns of a training matrix, allowing us to utilize the proposed schemes such as (P11.5) to learn an adaptive transform for patches.

Similar to the previous formulation [2] for adaptive square transform-based denoising, we propose the following image denoising problem formulation that exploits the union of transforms model.

$$\begin{aligned} \min_{\{W_k, x_i, \alpha_i, C_k\}} & \sum_{k=1}^{K} \sum_{i \in C_k} \left\{ \|W_k x_i - \alpha_i\|_2^2 + \lambda_i' \, Q(W_k) \right\} \\ & + \tau \sum_{i=1}^{N} \|R_i \, y - x_i\|_2^2 \\ s.t. \;\; & \|\alpha_i\|_0 \leq s_i \;\; \forall \; i, \;\; \{C_k\} \in G \qquad \text{(P11.8)} \end{aligned}$$

Here, $R_i \in \mathbb{R}^{n \times P}$ is defined to be a patch extraction operator, i.e., $R_i y \in \mathbb{R}^n$ denotes the i^{th} patch of the image y as a vector. We assume a total of N overlapping patches. Compared with Problem (P11.5), the denoising problem includes the additional, yet important data fidelity term $\tau \sum_{i=1}^{N} \|R_i \, y - x_i\|_2^2$. The assumption in (P11.8) is that there exist noiseless $x_i \in \mathbb{R}^n$ that approximate $R_i y$, and are approximately sparsifiable by the learned model. The weight τ for the fidelity term is typically inversely proportional to the noise level σ, that is assumed known a priori. Vector $\alpha_i \in \mathbb{R}^n$ in (P11.8) denotes the sparse representation of x_i in a specific cluster transform W_k, with an a priori unknown sparsity level s_i. The weight λ_i' is set based on the given noisy data $R_i y$ as $\lambda_0 \|R_i y\|_2^2$. The net weight on the $Q(W_k)$ regularizer in (P11.8) is then $\lambda_k = \sum_{i \in C_k} \lambda_i'$. Thus, similar to Problem (P11.5), the weight λ_k here varies depending on C_k.

Since $\tau \propto 1/\sigma$, we have the result that when $\sigma \to 0$, the optimal $x_i \to R_i y$ in (P11.8). In the limit, (P11.8) reduces to the transform learning problem (P11.5). Since the patch-based framework is used in formulation (P11.8), the denoised image x is obtained by averaging the learned x_i's at their respective locations in the image [2].

Problem (P11.8) has the disadvantage that it involves a priori unknown sparsity levels s_i. These sparsity levels have to be estimated in practice. An alternative version of (P11.8) would replace the penalty $\tau \sum_{i=1}^{N} \|R_i\, y - x_i\|_2^2$ by constraints $\|R_i\, y - x_i\|_2^2 \leq nC^2\sigma^2 \; \forall\, i$, where C is a constant. Furthermore, the sparsity constraints in (P11.8) can be converted to a penalty of the form $\sum_{i=1}^{N} \gamma_i \, \|\alpha_i\|_0$. Although this modification eliminates the issue of unknown sparsity levels s_i, it introduces another new set of unknown parameters $\gamma_i > 0$ [18]. In this chapter, we will work with the formulation (P11.8) that uses the (simple) data fidelity penalty and sparsity constraints. Our algorithm for (P11.8) will additionally estimate the (minimum) sparsity levels for which the condition $\|R_i\, y - x_i\|_2^2 \leq nC^2\sigma^2 \; \forall \; i$ is satisfied (similar to [1]).

11.5.2 Algorithm for (P11.8)

The proposed iterative algorithm is aimed at solving the non-convex Problem (P11.8). While one could solve (P11.8) with fixed s_i (e.g., s_i set to 10% of the patch size), we observed that the denoising performance is better when the s_i's are tuned adaptively as discussed above. Each iteration of our algorithm involves intra-cluster transform learning, variable sparsity level update, and clustering steps. The denoised patches x_i are updated in the final iteration. The denoised image is reconstructed when the iterations complete.

11.5.2.1 Intra-Cluster Transform Learning

Given $\{x_i\}$, $\{s_i\}$, and the clusters C_k, we solve for the cluster transforms $\{W_k\}$ and the corresponding sparse codes $\{\alpha_i\}$ in (P11.8). This problem separates out into K different single transform learning problems (similar to (P11.3)). The kth problem is as follows.

$$\begin{aligned} \min_{\{W_k, \alpha_i\}} \sum_{i \in C_k} & \left\{ \|W_k x_i - \alpha_i\|_2^2 + \lambda_i' \, Q(W_k) \right\} \\ s.t. \;\; & \|\alpha_i\|_0 \leq s_i \;\; \forall \; i \in C_k \end{aligned} \tag{11.49}$$

This problem is solved by alternating between sparse coding and transform update steps. For each of these steps, we use closed-form solutions [6].

[18] The γ_i's need to be set accurately for the modified formulation to work well in practice.

11.5.2.2 Intra-Cluster Sparsity Level Update

Now, we update the sparsity levels s_i for all i. We adopt a similar method for updating the sparsity levels as introduced in our prior work [2].

With a fixed cluster transform W_k and α_i ($i \in C_k$), one can solve for x_i in (P11.8) in the least squares sense as follows.

$$x_i = \begin{bmatrix} \sqrt{\tau}\, I \\ W_k \end{bmatrix}^{\dagger} \begin{bmatrix} \sqrt{\tau} R_i y \\ \alpha_i \end{bmatrix} = G_1 R_i y + G_2 \alpha_i \tag{11.50}$$

where I is the $n \times n$ identity, and the matrices G_1 and G_2 are given as $G_1 = \tau \left(\tau I + W_k^T W_k\right)^{-1}$ and $G_2 = \left(\tau I + W_k^T W_k\right)^{-1} W_k^T$. Both G_1 and G_2 are computed once for each cluster.

With α_i held at $H_{s_i}(W_k R_i y)$, we choose s_i to be the smallest integer such that the x_i in (11.50) satisfies the error condition $\|R_i\, y - x_i\|_2^2 \leq nC^2\sigma^2$. This can be done efficiently by pre-computing $m_i = G_1 R_i y$ and adding to it one scaled column of G_2 at a time (corresponding to incrementing s_i by 1 in $\alpha_i = H_{s_i}(W_k R_i y)$), until the error condition is met.

We only update the s_i's in this step, except in the final algorithm iteration, when the x_i's computed above satisfying the $\|R_i\, y - x_i\|_2^2 \leq nC^2\sigma^2$ condition represent the final denoised patches. Note that the sparse code is also further updated here for each $i \in C_k$ as $\alpha_i = H_{s_i}(W_k R_i y)$, using the optimal s_i.

11.5.2.3 Clustering

With fixed $\{s_i\}$, $\{W_k\}$, and $\{x_i\}$, we solve (P11.8) with respect to the clusters $\{C_k\}$ and sparse codes $\{\alpha_i\}$. This problem is similar to (P11.6). For each i, we solve a sparse coding and clustering problem in the union of transforms model. We calculate $\tilde{\alpha}_i^k = H_{s_i}(W_k R_i y)\ \forall\, k$, and choose the cluster $C_{\hat{k}_i}$ if we have that $\left\|W_{\hat{k}_i} x_i - \tilde{\alpha}_i^{\hat{k}_i}\right\|_2^2 + \eta_{\hat{k}_i} \|x_i\|_2^2 \leq \left\|W_j x_i - \tilde{\alpha}_i^j\right\|_2^2 + \eta_j \|x_i\|_2^2\ \ \forall\, j \neq \hat{k}_i$, where $\eta_j = \lambda_0\, Q(W_j)$. Then, the optimal $\alpha_i = \tilde{\alpha}_i^{\hat{k}_i}$. Note that the clustering step is not performed in the final iteration of our algorithm for (P11.8).

11.5.2.4 Computing Denoised Image Estimate

The denoised image patches $\{x_i\}$ obtained from the iterative scheme for (P11.8) are restricted to their range (e.g., 0-255 for unsigned 8-bit integer class). We output the denoised image by averaging the denoised patches at their respective image locations. The summary of the method for (P11.8) is presented in Fig. 11.2.

In order to enhance the method's efficiency, we typically perform the intra-cluster transform learning in our algorithm using only a subset of patches that are selected uniformly at random in each cluster. The patches are all mean-subtracted in our algorithm, and the means are added back to the final denoised patch estimates.

Our algorithm learns a union of transforms using noisy patches, and updates the sparsity levels s_i adaptively during the iterations. One could use the final s_i's output from the algorithm and re-solve (P11.8) with fixed s_i's by alternating between the intra-cluster transform learning, x_i update (by least squares), and clustering steps. However, we observed in our experiments that such additional iterations produce at most a minor additional improvement in denoising performance. Hence, to save run time, we do not include them.

Note that similar to previous work [2], we do not enforce the constraint $R_i x = x_i \ \forall\ i$ explicitly in (P11.8) [19], but rather treat extracted patches as individual data points. Although the final denoised image estimate is computed by averaging all patches (at respective locations), the approach may be sub-optimal [8], but results in an effective algorithm with low computational cost. Numerical results presented in Section 11.6 demonstrate this.

[19]Similar to the observation in Section 3.4 of Chapter 3, the data fidelity penalty $\sum_{i=1}^{N} \|R_i\, y - x_i\|_2^2$ in the objective of (P11.8) becomes a scaled log-likelihood with the explicit constraint $R_i x = x_i \ \forall\ i$, and assuming that all overlapping patches (i.e., periodically positioned 2D image patches, with 1 pixel shifts between them) including'wrap around' patches [5] (i.e., patches that begin near the right or bottom edges of an image are allowed to wrap around on the other side of the image (to complete them)) are included in our formulation. In that case, the data fidelity penalty is $\sum_{i=1}^{N} \|R_i\, y - x_i\|_2^2 = \sum_{i=1}^{N} \|R_i\, y - R_i x\|_2^2 = n\, \|y - x\|_2^2$, which is just a scaled version of the log-likelihood (with the likelihood denoted as $p(y|x)$) for the case when the noise in the pixels of y is i.i.d. Gaussian (so $p(y|x)$ is a Gaussian density function). Although we do not enforce the constraint $R_i x = x_i \ \forall\ i$ directly in (P11.8), we do obtain the denoised x in the end as the least-squares solution in the set of equations $R_i x = x_i \ \forall\ i$ (with the x_i's here being the denoised patches).

```
Algorithm for (P11.8)
Input :   $y$ - noisy image, $s$ - initial fixed sparsity, $K$ - number of clusters, $L$ - number of iterations of algorithm for (P11.8), $\sigma^2$ - an estimate of noise variance, $J$ - number of transform learning iterations, $\lambda_0$ - a constant.
Output :   $x$ - Denoised image estimate
Initialization :   Patches $x_i = R_i y$ and $s_i = s$ for $i = 1, 2, ...., N$. Initial $W_k = W_0 \, \forall \, k$, $C_k$ - random cluster selection for each $i \in \{1, ..., N\}$.
For l = 1:L Repeat
    1. For $k = 1...K$, update $W_k$ and the corresponding $\alpha_i$ alternatingly (to solve problem (11.49)), with fixed clusters $C_k$ and $x_i = R_i y$. The number of alternations for each $k$ is $J$.
    2. Update $s_i$ for all $i = 1, 2, ...., N$ :  Increase $s_i$ in $\alpha_i = H_{s_i}(W_k R_i y)$ in (11.50) where $i \in C_k$, until the error condition $\|R_i \, y - x_i\|_2^2 \leq nC^2\sigma^2$ is reached.
    3. For each $i \in \{1, ..., N\}$, perform clustering and sparse coding with $x_i = R_i y$: calculate $\tilde{\alpha}_i^k = H_{s_i}(W_k x_i) \; \forall \, k$ and assign $i$ to cluster $C_{\hat{k}}$ if $\hat{k}$ is the smallest integer in $\{1, ..., K\}$ such that $\left\| W_{\hat{k}} x_i - \tilde{\alpha}_i^{\hat{k}} \right\|_2^2 + \eta_{\hat{k}} \|x_i\|_2^2 \leq \left\| W_j x_i - \tilde{\alpha}_i^j \right\|_2^2 + \eta_j \|x_i\|_2^2 \;\; \forall \, j \neq \hat{k}$ holds with $\eta_j = \lambda_0 \, Q(W_j)$ $\forall \; j$. The optimal code $\alpha_i = \tilde{\alpha}_i^{\hat{k}}$.
End
Update $x$ : Obtain the denoised patches $x_i$ satisfying the error condition in step 2 above, and average them at their respective image locations.
```

Figure 11.2: Algorithm to solve (P11.8), and obtain a denoised image estimate x. A particular initialization is mentioned above for $\{W_k, C_k, s_i\}$, for simplicity. The W_0 above can be chosen to be the DCT, Wavelets, etc.

11.5.3 Improved Denoising by Iteratively Resolving (P11.8)

Our aforementioned algorithm obtains a denoised image estimate by solving (P11.8) once. We propose an improved iterative denoising scheme that makes multiple passes through (P11.8), each time replacing y by its latest denoised version, setting the noise level to an estimate of the remaining noise in the denoised image produced in the previous pass. Each iteration of this denoising scheme uses the same algorithm (for (P11.8)) as in Section 11.5.2.

11.6 Numerical Experiments

11.6.1 Framework

11.6.1.1 Overview

In this section, we present results demonstrating the promise of the proposed adaptive union of transforms, or OCTOBOS framework in representing, or denoising images. First, we illustrate the convergence behavior of our alternating transform learning algorithm. We consider the behavior of our algorithm with various initializations to empirically check whether the algorithm is sensitive to initializations. We will also provide examples showing the clustering/classification ability of our approach. Then, we indicate the promise of the proposed transform learning approach for representing images. Finally, we demonstrate the potential of the proposed adaptive OCTOBOS transform-based image denoising scheme. We will show that the proposed approach performs better than methods involving learned single square transforms or learned overcomplete synthesis dictionaries (K-SVD [20]). The computational advantage of the proposed approach over the synthesis dictionary-based approach will also be indicated. Furthermore, we demonstrate that our method denoises better than GMM denoising [8], and is competitive with the state-of-the-art BM3D [7] for some images and/or noise levels.

The data in our experiments are generated as the patches of natural images. We employ our proposed transform learning Problem formulation (P11.5) for learning adaptive sparse representations of such patches. The means (DC values) of the patches are removed and we only sparsify the mean-subtracted patches which, after reshaping as vectors, are stacked as columns of the training matrix Y. The means are added back for image display.

[20]The K-SVD method is a highly popular scheme that has been applied to a wide variety of image processing applications [1, 56]. Mairal et al. [194] have proposed a non-local method for denoising, that also exploits learned dictionaries. A similar extension of OCTOBOS learning-based denoising using non-local means methodology may potentially provide enhanced performance for OCTOBOS. However, such an extension would distract from the focus on the OCTOBOS model in this work. Hence, we leave its investigation for future work. For the sake of simplicity, we compare our overcomplete transform learning scheme to the corresponding overcomplete synthesis dictionary learning scheme K-SVD in this work.

Mean removal is typically adopted in image processing applications such as compression and image denoising [86, 2].

All our (unoptimized) implementations were coded in Matlab version R2013b. The corresponding Matlab implementation of K-SVD denoising [1] available from Elad's website [86] was used in our comparisons. For K-SVD denoising, we used the built-in parameter settings of the author's implementation. We also used the publicly available implementations of GMM [195] and BM3D denoising [196]. All computations were performed with an Intel Core i7 CPU at 2.9GHz and 4GB memory, employing a 64-bit Windows 7 operating system.

11.6.1.2 Quality/Performance Metrics:

Several metrics have been introduced previously to evaluate the quality of learned transforms [9, 2]. The *normalized sparsification error* (NSE) for a single transform W is defined as $\|WY - X\|_F^2 / \ \|WY\|_F^2$, where Y is the data matrix, and the columns $X_i = H_s(WY_i)$ of the matrix X denote the sparse codes. The NSE measures the fraction of energy lost in sparse fitting in the transform domain, and is an interesting property to observe for the adaptive transforms. For our proposed approach, since we have a union of square transforms and clustered patches, we compute the normalized sparsification error as follows.

$$\text{NSE} = \frac{\sum_{k=1}^{K} \sum_{i \in C_k} \|W_k Y_i - X_i\|_2^2}{\sum_{k=1}^{K} \sum_{i \in C_k} \|W_k Y_i\|_2^2} \tag{11.51}$$

Here, the numerator is the net sparsification error (i.e., the total transform domain residual), and the denominator denotes the total transform domain energy. We have $0 \leq \text{NSE} \leq 1$. The sparse codes in the k^{th} cluster above are $X_i = H_s(W_k Y_i)$. For the proposed NSE definition to be meaningful, we assume that the W_k's are all normalized (e.g., they have unit spectral norm). When $K = 1$, the above definition is identical to the previously proposed NSE [9] for a single transform.

For image representation, a useful performance metric is the recovery peak signal to noise ratio (or *recovery PSNR* (rPSNR)), which for the case of a single transform W was previously defined as $20 \log_{10}\left(255\sqrt{P} / \ \|Y - W^{-1}X\|_F\right)$ in decibels (dB), where P is the number of image pixels and X is again the transform sparse code of data Y. The recovery PSNR measures the error in recovering the patches Y (or equivalently the image, in the case of non-overlapping patches) as $W^{-1}X$ from their sparse codes X obtained by projecting WY onto the ℓ_0 ball. The recovery PSNR serves as a simple surrogate for the performance of a learned transform in a compression application. For our proposed union of transforms approach, the recovery PSNR is redefined in terms of the clusters as follows.

$$\text{rPSNR} = 20\log_{10}\left(\frac{255\sqrt{P}}{\sqrt{\sum_{k=1}^{K}\sum_{i\in C_k}\left\|Y_i - W_k^{-1}X_i\right\|_2^2}}\right) \tag{11.52}$$

Note that each patch Y_i belongs to exactly one cluster C_k above.

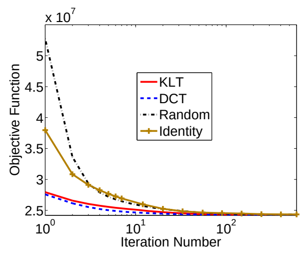
For image denoising, similar to previous work [1], we measure the image reconstruction PSNR computed between the true noiseless reference and the noisy or denoised images.

11.6.2 Convergence and Learning Behavior

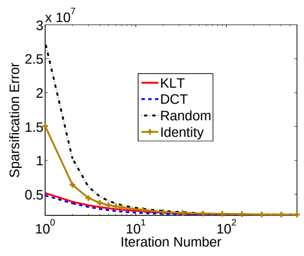
Here, we illustrate the convergence behavior of our alternating OCTOBOS learning algorithm for image data. We extract the $\sqrt{n}\times\sqrt{n} = 8\times 8$ ($n = 64$) non-overlapping mean-subtracted patches from the 512×512 Barbara image (shown later in Fig. 11.8). The data matrix $Y \in \mathbb{R}^{n\times N}$ in this case has 4096 vectorized training patches. We learn a union of transforms (or, equivalently an OCTOBOS transform) for this data by solving (P11.5). The parameters are set as $\lambda_0 = 3.1\times 10^{-3}$, $s = 11$ (which is roughly $1/6^{\text{th}}$ of the data dimension), and the number of clusters $K = 2$. The choice of λ_0 here ensures well-conditioning of the blocks of the learned overcomplete transform. Badly conditioned transforms degrade performance in applications [9]. Hence, we focus our investigations here only on the well-conditioned scenario.

In the experiments of this chapter, we assume an initialization (or, initial estimates) for the clusters $\{C_k\}$ and cluster transforms $\{W_k\}$ in (P11.5). The initial sparse codes in (P11.5) are then computed for the initialization as $X_i = H_s(W_kY_i)$, $\forall i \in C_k$, and for each k, and the alternating Algorithm A1 is executed beginning with the transform update step. Note that the initial $\{X_i\}$ are fully determined by the initial estimates for the cluster-specific transforms.

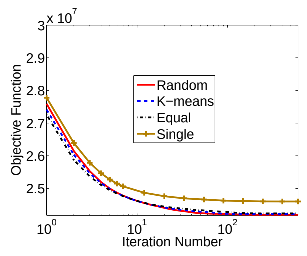
Here, we study the convergence behavior of the algorithm for various ini-



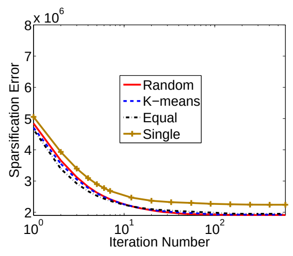
(a)



(b)

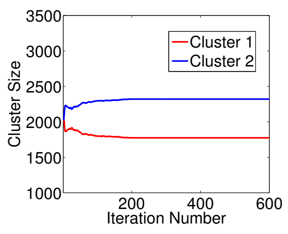


(c)

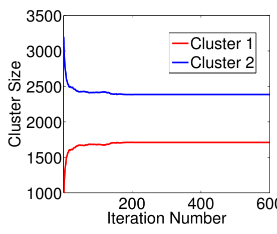


(d)

Figure 11.3: Behavior of the OCTOBOS Learning Algorithm for (P11.5): (a) Objective function with different transform initializations; (b) Sparsification error with different transform initializations; (c) Objective function with different cluster initializations for $K = 2$, along with the objective for single square transform ($K = 1$) learning; (d) Sparsification error with different cluster initializations for $K = 2$, along with the sparsification error for single square transform ($K = 1$) learning.



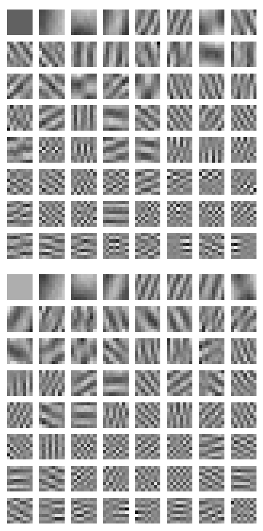
(a)



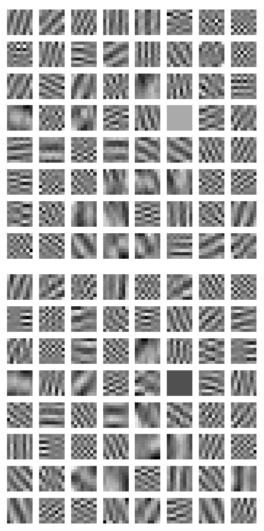
(b)

Figure 11.4: Cluster size convergence for (P11.5) corresponding to Fig. 11.3(c): (a) Case of random cluster initialization; (b) Case of 'equal' cluster initialization. Note that the cluster sizes change dramatically in the first iteration in (b).

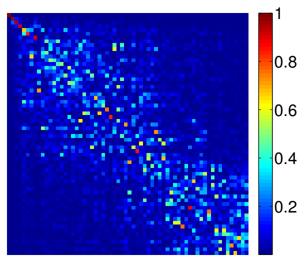
tializations. We consider two different scenarios. In Scenario A, we fix the initial clusters $\{C_k\}$ (each patch is assigned uniformly at random to one of $K = 2$ clusters), and vary the initialization for the cluster transforms $\{W_k\}$. Four different initializations for the $\{W_k\}$ are considered: (i) the 64×64



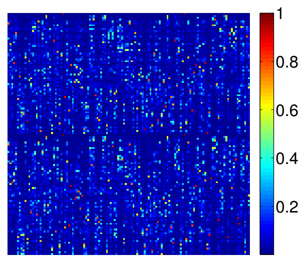
(a)



(b)



(c)



(d)

Figure 11.5: Learned OCTOBOS transforms corresponding to Fig. 11.3(a): Rows of the learned overcomplete transform W shown as patches (the two square blocks of W are separated by a white space) for the case of (a) KLT initialization, and (b) random matrix initialization; Magnitude of the cross-gram matrix computed: (c) between the two learned (row-normalized) square blocks in (a); and (d) between the two (row-normalized) overcomplete transforms in (a) and (b).

2D DCT matrix (obtained as the Kronecker product of two 8×8 1D DCT matrices); (ii) the Karhunen-Loève Transform (KLT) initialization, obtained by inverting (transposing) the left singular matrices of the data in each cluster; (iii) the identity matrix; and (iv) a random matrix with i.i.d. Gaussian entries (zero mean and standard deviation 0.2), respectively [21]. In Scenario B, we fix the initial cluster transforms $\{W_k\}$ to be the 2D DCT, and vary the initialization for the $\{C_k\}$ in (P11.5). We consider three initializations

[21]Note that for the case of the DCT, identity, and random initializations, the same matrix is used to initialize all the W_k's.

for the clusters here: (i) the initialization obtained by using the well-known k-means algorithm; (ii) random clustering, where each patch is assigned uniformly at random (a different random clustering is used here, than the one in the aforementioned Scenario A) to one of the clusters; and (iii) the patches on the left half of the image in one cluster, and the remaining patches in the second cluster. We will refer to the initialization (iii) as 'equal' initialization, for simplicity.

Figure 11.3 shows the progress of the algorithm over iterations for the various initializations of $\{W_k\}$ (Scenario A in Figs. 11.3(a) and 11.3(b)), and $\{C_k\}$ (Scenario B in Figs. 11.3(c) and 11.3(d)). Both the objective function (Figs. 11.3(a) and 11.3(c)) and sparsification error (Figs. 11.3(b) and 11.3(d)) converge quickly for our algorithm. Importantly, the final values of the objective (similarly, the sparsification error) are nearly identical for all the initializations. This indicates that our learning algorithm is reasonably robust, or insensitive to initialization. Good initializations for the $\{W_k\}$ such as the DCT and KLT lead to faster convergence of learning (Figs. 11.3(a) and 11.3(b)).

For comparison, we also plot in Figs. 11.3(c) and 11.3(d), the behavior of the algorithm for $K = 1$. In this case it reduces to the single square transform learning algorithm via (P11.3) [9, 6]. The parameters such as s and λ_0 are set to the same values as for $K = 2$. Fig. 11.3(c) shows the objective for single square transform learning converging to a larger value compared to OCTOBOS learning. Likewise, the sparsification error for OCTOBOS for $K = 2$ (Fig. 11.3(d)) is 0.67 dB better than that provided by the learned single square transform. This confirms our expectation based on Proposition 17 in Section 11.2.

The learned square blocks of the overcomplete transform here have similar condition numbers (≈ 1.4) and Frobenius norms (≈ 5) for all initializations. This confirms that an appropriate choice of λ_0 allows the learned W_k's to be similarly scaled, and ensures that the sparsification error term in (P11.5) is fully meaningful.

Next, we plot in Fig. 11.4, the cluster sizes over iterations for two different initializations of $\{C_k\}$ – random (Fig. 11.4(a)) and 'equal' (Fig. 11.4(b)). The final values of $|C_1|$ (alternatively $|C_2|$) for the two initializations are similar. We observed that the (learned) clusters themselves can be similar (although, not necessarily identical) for various initializations.

Figure 11.5 visualizes the transforms learned by our alternating algorithm with the KLT and with random initializations for $\{W_k\}$ (the aforementioned Scenario A). The rows, or atoms of the learned overcomplete transforms are shown as patches. The learned transforms exhibit geometric and texture-like features, achieved by adaptation to the patches of the Barbara image. In order to gauge the similarity, or difference between the learned OCTOBOS transforms with different initializations, we show in Figure 11.5(d), the magnitude of the cross-gram matrix [22] computed between the transforms in Figs. 11.5(a) and 11.5(b). We normalize the rows of the transforms prior to computing their cross-gram matrix. The cross-gram matrix then indicates the coherence between every pair of rows from the two overcomplete transforms. For the 128×128 cross-gram matrix in this case, there are only 15 entries with amplitude above 0.9. This indicates that the two learned OCTOBOS transforms are not similar, i.e., they are not related by just row permutations and sign changes. However, interestingly, both still sparsify the data Y equally well. Therefore, as far as sparsification is concerned, the two different overcomplete transforms can be considered essentially equivalent [9].

How similar are the square blocks of the same overcomplete transform? In Fig. 11.5(c), we show the magnitude of the 64×64 cross-gram matrix computed between the (row normalized) blocks in Fig. 11.5(a). In this case, there are only 5 entries with amplitude above 0.9, indicating that the two learned square blocks are quite different. This is not surprising, since the two blocks here correspond to disjoint clusters.

Although we considered the image Barbara in our convergence study here, we observed similar behavior for our algorithm for other images as well.

11.6.3 Clustering Behavior

In this subsection, we briefly illustrate the clustering behavior of our OCTOBOS learning scheme. First, we consider the 251×249 input image shown in Fig. 11.6(a). The image was formed by combining two textures from the Brodatz database [197, 198]. The goal is to cluster the pixels of the image into one of two classes. In order to do so, we adopt the following strategy. We consider all overlapping mean-subtracted patches from the input image, and

[22]For two matrices A and B of same size, the cross-gram matrix is computed as AB^T.

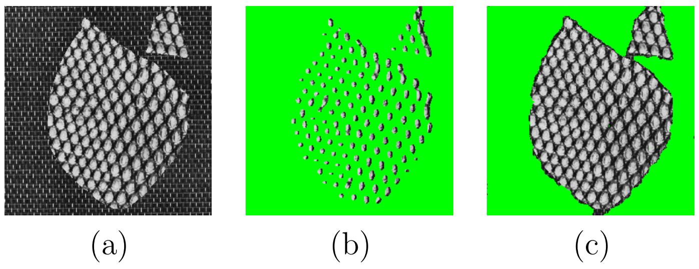


Figure 11.6: $K = 2$ clustering example: (a) Input image. (b) Input image with pixels clustered into Class 1 shown in Green for the K-means initialization, and (c) OCTOBOS.

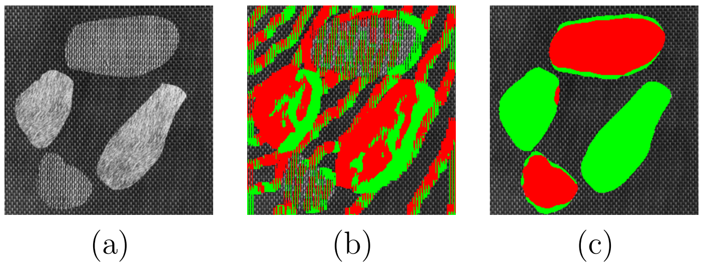


Figure 11.7: $K = 3$ clustering example: (a) Input image, (b) Input image with pixels clustered into Classes 1 and 2 shown in Green and Red, respectively, for the K-means initialization, (c) Input image with pixels clustered into Classes 1 and 2 shown in Green and Red, respectively, for OCTOBOS.

employ formulation (P11.5) to learn an adaptive clustering of the patches. Since overlapping patches are used, each pixel in the image typically belongs to many overlapping patches. We cluster a pixel into a particular class C_k if the majority of the patches to which it belongs, are clustered into that class by (P11.5).

We use 9×9 (overlapping mean-subtracted) patches ($n = 81$), and set $s = 10$, $K = 2$, and λ_0 is set as in Section 11.6.2. We initialize OCTOBOS learning using the clustering result of the k-means algorithm. The two cluster transforms are initialized with the DCT. We now use the aforementioned strategy to cluster the pixels of the input two-texture image into one of two classes using OCTOBOS. Fig. 11.6(c) shows the clustering result obtained using OCTOBOS. As a comparison, Fig. 11.6(b) shows the image pixels clustered into each class for the k-means initialization. The proposed scheme is seen to improve over the k-means result. Alternative initializations for the

OCTOBOS clusters such as random initialization also provide similar final clustering results, but typically require more iterations to converge.

Fig. 11.7 shows clustering results for a 256×256 three texture image (Fig. 11.7(a)). The parameters for OCTOBOS are $K = 3$, and n, s, λ_0 are set just as for the case of Fig. 11.6. OCTOBOS (Fig. 11.7(c)) is again seen to improve over the k-means initialization (Fig. 11.7(b)).

The clustering examples here illustrate some *preliminary* potential for the OCTOBOS scheme in classification. We also observed reasonable clustering results with other texture images. Note that unlike prior work in synthesis dictionary-based classification (e.g., [179]), we do not have additional penalties in (P11.5) that discriminate (e.g., by enforcing incoherence) between the learned transform blocks. An extension of our OCTOBOS scheme by incorporating such classification-specific penalties (and other classification-specific heuristics) may be useful for the classification application. We leave the detailed investigation of the classification application (for example, the study of potential discriminative OCTOBOS learning methods) for future work.

11.6.4 Sparse Representation of Images

We have studied the potential of the proposed OCTOBOS learning scheme for sparse representation of several images in [134]. The NSE and recovery PSNR metrics of Section 11.6.1 were used to measure the quality of the learned OCTOBOS transforms. The learned OCTOBOS transforms were shown to provide significantly better sparsification and recovery compared to analytical transforms such as the DCT, or even KLT. Importantly, as K increased, the learned OCTOBOS transforms provided increasingly better image representation compared to the learned square transform (cf. [134] for the complete details).

11.6.5 Image Denoising

We present preliminary results for our adaptive OCTOBOS-based image denoising framework (based on (P11.8)). We work with the six images shown in Fig. 11.8, and simulate i.i.d. Gaussian noise at 5 different noise levels ($\sigma = 5$,

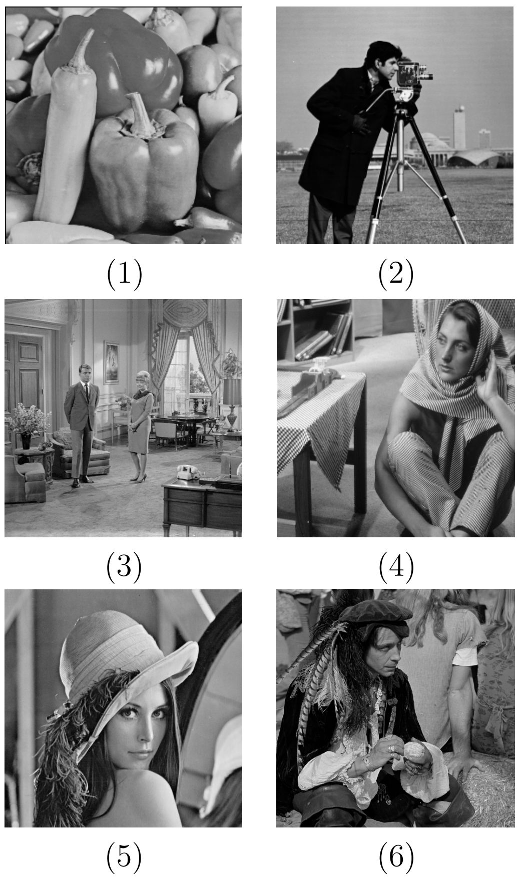


Figure 11.8: Images used for denoising experiments: (1) Peppers, (2) Cameraman, (3) Couple, (4) Barbara, (5) Lena, and (6) Man.

10, 15, 20, 100) for each of the images. We compare the denoising results obtained by our proposed algorithm in Section 11.5, with those obtained by the adaptive overcomplete K-SVD denoising scheme [1], the GMM-based denoising method [8], and the BM3D method [7], which is a state-of-the-art image denoising method. Note that as opposed to the K-SVD scheme, our OCTOBOS method is quite constrained due to the block cosparsity of the sparse code.

We work with 8×8 ($n = 64$) overlapping image patches in our experiments. For OCTOBOS-based denoising, we consider a 256×64 transform, i.e., $K = 4$. A corresponding 64×256 synthesis dictionary is used in the synthesis K-SVD denoising method. We fixed the initial sparsity levels s_i to 6 for all patches in our algorithm for (P11.8). We chose $C = 1.08$, and $\lambda_0 = 3.1 \times 10^{-2}$. We perform multiple passes through (P11.8), as discussed in Section 11.5.3.

For each noise level (σ) of the original noisy image, the number of times that (P11.8) is solved, and the corresponding noise levels (for each pass through (P11.8)) were determined empirically [23]. These same parameters were used for all the images in our experiments. Other parameters in our algorithm such as the number of iterations (L, J in Fig. 11.2) for (P11.8) were set empirically. An example of OCTOBOS denoising is shown in Fig. 11.9.

Table 11.2 lists the PSNRs obtained by denoising with OCTOBOS, overcomplete K-SVD, GMM, and BM3D. First, the OCTOBOS scheme clearly provides better PSNRs than K-SVD for all images and noise levels. Comparing the PSNR values obtained by the 256×64 OCTOBOS to those of the 64×256 synthesis K-SVD dictionary for each image and noise level, we obtain an average PSNR improvement (average computed over all images and noise levels) of 0.30 dB for OCTOBOS over K-SVD. The improvement over K-SVD for individual examples is up to 0.66 dB in Table 11.2. Thus, the OCTOBOS method outperforms K-SVD despite using a constrained (block cosparse) transform. We also obtain an average speedup of 2.8x for OCTOBOS denoising over K-SVD denoising [24]. This is because the various steps of OCTOBOS-based denoising such as the sparse coding and clustering step are computationally very cheap.

Our OCTOBOS denoising scheme is also 0.05 dB better on an average (over all images and noise levels) compared to GMM-based denoising in Table 11.2. Although the state-of-the-art BM3D method is quite better than OCTOBOS at $\sigma = 100$, OCTOBOS denoising is only 0.22 dB worse than BM3D on the average at other noise levels ($\sigma \leq 20$ in Table 11.2). OCTOBOS denoising also performs comparably to BM3D (at lower noise levels) for certain images such as Cameraman and Peppers.

Next, using the same parameters as in the preceding experiments, we study the behavior of OCTOBOS denoising as a function of the overcompleteness K of the transform. Figs. 11.10(a) and 11.10(b) plot the denoising PSNRs for Barbara as a function of the number of clusters K for $\sigma = 10$ and $\sigma = 20$,

[23]The noise level estimates decrease over the iterations (passes through (P11.8)). We also found empirically that underestimating the noise standard deviation (during each pass through (P11.8)) led to better performance.

[24]Our MATLAB implementation of OCTOBOS denoising is not currently optimized for efficiency. Therefore, the speedup here is computed by comparing our unoptimized MATLAB implementation to the corresponding MATLAB implementation [86] of K-SVD denoising.

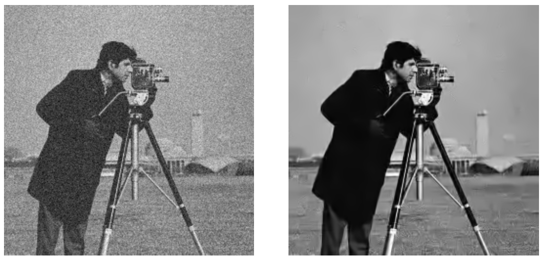
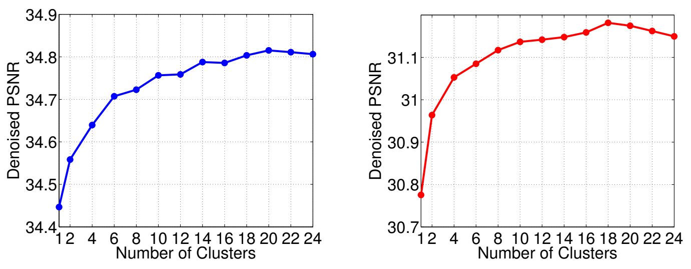

(a) (b)

Figure 11.9: Denoising result: (a) Noisy Cameraman (PSNR = 22.10 dB), (b) Denoised Cameraman (PSNR = 30.24 dB) obtained using the OCTOBOS scheme.

respectively. In both cases, the denoising PSNR increases with K up to an optimal value of K, beyond which the PSNR begins to slowly drop. Initially, as K increases, the OCTOBOS model becomes richer, and thus, provides increasingly better denoising. However, when K becomes too large [25], one cannot reliably learn all the OCTOBOS square blocks from the limited number of noisy training data associated with each block, without overfitting the noise. Thus, the PSNR begins to drop for very large K. This effect is more pronounced the higher the noise level, as seen in Fig. 11.10, where the optimal K where the plot peaks is lower for $\sigma = 20$, than for $\sigma = 10$. The same trend continues at $\sigma = 100$ (not shown in Fig. 11.10). The plots in Fig. 11.10 also illustrate the advantage (up to 0.4 dB improvement for this example) of OCTOBOS-based denoising over the single square transform-based ($K = 1$) denoising. This gap increases when the OCTOBOS parameters are better tuned for larger K.

Thus, our results for OCTOBOS-based denoising are quite comparable to, or better than the results obtained by previous image denoising schemes such as GMM denoising, BM3D, K-SVD denoising, and adaptive square transform denoising. The learned OCTOBOS (square) blocks for all images and noise levels in our experiments, are well-conditioned (condition numbers of $1-2$). We expect the denoising PSNRs for OCTOBOS to improve further with optimal parameter tuning. Our method is limited at very high noise (such as $\sigma = 100$) due to the fact that the learning is done using corrupted data. Therefore, in the high noise setting, using a fixed OCTOBOS transform

[25]Compare this behavior to the monotone increase with K of the recovery PSNR for image representation – see [134].



(a) (b)

Figure 11.10: Denoising PSNR for Barbara as a function of the number of clusters K: (a) $\sigma = 10$, (b) $\sigma = 20$.

(learned over a database of images that share similar properties to the image being denoised) may provide better denoising. This topic is worthy of further future investigation. Moreover, since the state-of-the-art BM3D is a non-local method, we believe that a non-local extension to the OCTOBOS scheme could lead to even better OCTOBOS denoising performance. We plan to investigate such an extension in the near future.

11.7 Conclusions

In this chapter, focusing on the transform model for sparse representations, we presented a novel union of sparsifying transforms model. We showed that this model can also be interpreted as an overcomplete sparsifying transform model with an extra block cosparsity constraint (OCTOBOS) on the sparse code. The sparse coding in the proposed model can be interpreted as a form of clustering. We presented a novel problem formulation and algorithm for learning the proposed OCTOBOS transforms. Our algorithm involves simple closed-form solutions, and is thus computationally very efficient. Our theoretical analysis established global convergence of the algorithm to the set of partial minimizers of the proposed non-convex learning problem. For natural images, our learning scheme gives rise to a union of well-conditioned transforms, and clustered (classified) patches or textures. It is also usually insensitive to initialization. The adapted model provides better sparsification errors and recovery PSNRs for images compared to learned single square transforms, and analytical transforms. In the application of image denoising, the proposed scheme typically provides better image reconstruction quality compared to adaptive (single) square transforms, and adaptive overcomplete synthesis dictionaries. These results suggest that the proposed OCTOBOS learning produces effective models adapted to the data.

Image	σ	Noisy PSNR	BM3D	K-SVD	GMM	OCTOBOS
Peppers	5	34.14	38.09	37.78	37.95	38.09
	10	28.10	34.66	34.24	34.51	34.57
	15	24.58	32.69	32.18	32.54	32.43
	20	22.12	31.33	30.80	31.18	30.97
	100	8.11	23.17	21.79	22.97	22.23
Cameraman	5	34.12	38.21	37.81	38.06	38.19
	10	28.14	34.15	33.72	34.00	34.15
	15	24.61	31.91	31.50	31.85	31.94
	20	22.10	30.37	29.82	30.21	30.24
	100	8.14	23.15	21.76	22.89	22.24
Couple	5	34.16	37.48	37.28	37.35	37.40
	10	28.11	34.01	33.51	33.79	33.73
	15	24.59	32.08	31.46	31.84	31.71
	20	22.11	30.78	30.02	30.51	30.34
	100	8.13	23.46	22.57	23.30	22.88
Barbara	5	34.15	38.30	38.08	37.59	38.31
	10	28.14	34.97	34.41	33.61	34.64
	15	24.59	33.05	32.33	31.28	32.53
	20	22.13	31.74	30.83	29.74	31.05
	100	8.11	23.61	21.87	22.13	22.41
Lena	5	34.16	38.70	38.61	38.55	38.71
	10	28.12	35.88	35.49	35.56	35.64
	15	24.63	34.26	33.74	33.87	33.92
	20	22.11	33.01	32.41	32.60	32.59
	100	8.14	25.75	24.51	25.24	25.17
Man	5	34.15	36.76	36.47	36.75	36.73
	10	28.13	33.18	32.71	33.14	32.98
	15	24.63	31.32	30.78	31.32	31.07
	20	22.11	30.03	29.40	30.02	29.74
	100	8.14	23.83	22.76	23.65	22.92

Table 11.2: PSNR values for denoising with 256×64 OCTOBOS transform, along with the corresponding values for denoising using BM3D [7], the 64×256 overcomplete K-SVD [1], and the GMM method [8]. The PSNR values of the noisy images are also shown.

CHAPTER 12

CONCLUSIONS AND FUTURE WORK

In this thesis, we studied various methods for the data-driven adaptation of sparsifying transforms. We list our various conclusions and future directions here on a chapter-by-chapter basis.

First, in Chapter 2, a novel problem formulation for learning unstructured square sparsifying transforms was presented. The alternating algorithm for square transform learning involves two steps - thresholding and nonlinear conjugate gradients (NLCG). Our proposed framework gives rise to well-conditioned transforms with much lower sparsification errors than analytical transforms. Results with natural images demonstrate that well-conditioning (but not necessarily unit conditioning) of the transforms is compatible with good sparsification and good performance in applications. Even for piecewise constant images, for which a difference operator provides optimal sparsification, but at high condition number, our well-conditioned learned transforms provide essentially identical, or even better sparsification. Our algorithm was shown to provide monotonic convergence of the cost function, and is insensitive to initialization. Moreover, the per-iteration computational cost of our transform learning is nearly two orders of magnitude lower than that of synthesis dictionary learning algorithms such as K-SVD. We also introduced a signal denoising formulation involving sparsifying transform learning, and demonstrated promising performance for our proposed algorithm.

In Chapter 3, we presented novel problem formulations for learning square doubly sparse transforms. The proposed formulations give rise to significantly sparse and well-conditioned transforms with better sparsification errors and recovery PSNRs than analytical transforms. Moreover, imposing the doubly sparse property leads to faster learning and faster computations with the sparse transform. The adapted doubly sparse transform has a reduced storage requirement and generalizes better than the unstructured (or non-sparse) transform. We also introduced a novel problem formulation for image denoising and demonstrated the promise of adaptive transforms in this application, with results competitive with overcomplete K-SVD. Most importantly, denoising with doubly sparse transforms is much cheaper. The usefulness of doubly sparse transform learning in image compression and other applications merits further study.

In Chapter 4, we studied further the problem formulations and algorithms for learning unstructured and well-conditioned square sparsifying transforms. The proposed alternating algorithms are an improvement over those in Chapter 2, and involve efficient optimal updates. Specifically, the proposed solution for the transform update step achieves the global minimum in that step. In the limit of $\lambda \to \infty$, the proposed algorithms become orthonormal transform (or orthonormal synthesis dictionary) learning algorithms. Importantly, we provided convergence guarantees for the proposed transform learning schemes. We established that our alternating algorithms are globally convergent to the set of local minimizers of the non-convex transform learning problems. Our convergence guarantee does not rely on any restrictive assumptions. The learned transforms obtained using our schemes provide better representations than analytical ones such as the DCT for images. Our learning algorithms are also faster than the previous ones involving iterative NLCG in the transform update step. In the application of image denoising, our algorithm provides comparable or better performance compared to synthesis K-SVD (which has no convergence properties), while being faster.

In Chapter 5, we presented a novel sparsifying transform-based framework for general blind compressed sensing. Our formulations exploit the transform domain sparsity of overlapping image patches in 2D (or voxels in 3D). The proposed formulations are however highly nonconvex. Our alternating algorithms for solving the proposed problems involve highly efficient update steps. Importantly, our algorithms are guaranteed to converge to the critical points of the objectives defining the proposed formulations. These critical points are guaranteed to be at least partial global and local minimizers. We also studied the usefulness of sparsifying transform-based blind compressed sensing for magnetic resonance imaging. Our numerical examples showed the usefulness of the proposed schemes for MR image reconstruction from highly undersampled k-space data. Our approaches, while being highly efficient (10x faster than synthesis dictionary-based schemes for 2D images), also provide promising MR image reconstruction quality, that is better than the quality provided by leading compressed sensing MR image reconstruction schemes. The usefulness of the proposed blind compressed sensing methods in other inverse problems and imaging applications merits further study.

In Chapter 6, we introduced an adaptive sampling framework for compressed sensing MRI. This framework was also combined with an adaptive reconstruction framework. The iterative algorithm for sampling design utilizes fully sampled training image scans to adapt an initial undersampling pattern. The k-space errors of the image reconstructed from the undersampled k-space data are reduced in each iteration. Significant improvements in reconstruction PSNR were observed in both training and test images when using the adapted sampling pattern compared to the initial pattern. The proposed framework for sampling design is generic and can be combined with any reconstruction strategy. A more detailed study of the parameters involved in sampling design and a comparison to other works (e.g., that of Seeger et al. [147]) will be presented elsewhere. We also plan to study the performance of alternative choices for the k-space error weighting function.

In Chapter 7, we presented a novel problem formulation for online learning of square sparsifying transforms. The formulation is to sequentially update the sparsifying transform and sparse code for signals that arrive or are processed sequentially. The proposed algorithm involves a sparse coding step and a transform update step per signal. Each of these steps is implemented efficiently. We also presented a mini-batch version of our online algorithm that can handle blocks of signals at a time. The proposed schemes were shown to be computationally much cheaper (in terms of cost per signal) than online synthesis dictionary learning. In practice, the online/mini-batch sparsifying transform learning converges better/faster than batch mode (where all signals are considered simultaneously) transform learning. We presented experiments demonstrating the usefulness of online transform learning in big data sparse representation, and denoising. The topics of online learning of an overcomplete transform, and online video denoising will be considered in future work.

In Chapter 8, we analyzed the convergence behavior of the online sparsifying transform learning algorithms proposed in Chapter 7. We showed that our online transform learning algorithms are guaranteed to converge (almost surely) to the set of stationary points of the learning problem. Specifically, every accumulation point of the iterate sequence in our algorithm is a stationary point of the expected cost (the gradient of the cost $g(W)$ is zero at each point) with probability 1. Moreover, every accumulation point corresponds to the same expected cost ($g(W)$) value with probability 1. Our guarantee relies on few (easy to verify, handle) assumptions. Moreover, the result is for a highly non-convex problem, that is not simply biconvex. Further investigation of the local/global optimality of our scheme will be considered in future work.

In Chapter 9, we presented the first convex sparsifying (doubly sparse) transform learning formulation, and an algorithm guaranteeing fast convergence to a global optimum. We also presented a 'partially non-convex' formulation by replacing the ℓ_1 sparsity penalty on the sparse code with an ℓ_0 constraint. In this case, the proposed algorithm has a local convergence property. In practice, it is also typically insensitive to initialization. For comparison, we also included a 'fully non-convex' version of our doubly sparse convex formulation by replacing the ℓ_1 penalties on both the sparse code and transform with ℓ_0 constraints. All our proposed formulations give rise to significantly sparse and well-conditioned transforms with much lower sparsification errors and recovery PSNRs than analytical transforms. Importantly, our convex formulation performs (in image representation) only slightly worse than its non-convex (i.e., the 'partially non-convex', and 'fully non-convex') variants. The learned transforms obtained via our proposed formulations in this chapter perform comparably or somewhat worse than those learned using the non-convex (non-guaranteed) schemes of Chapter 3 in our (preliminary) experiments. This may be because we exploit a different set of (potentially more restrictive) properties (skew-symmetry) here than in Chapter 3. Nevertheless, we would like to emphasize that the schemes proposed in this chapter such as the convex or 'partially non-convex' schemes have the advantage of strong convergence guarantees. The usefulness of the proposed schemes in image denoising or other applications merits detailed study.

In Chapter 10, we introduced a novel problem formulation for learning unstructured overcomplete sparsifying transforms. The proposed alternating algorithm for transform learning involves an exact and cheap sparse coding step, and a transform update step that is performed by iterative methods (CG). The learned transforms have better properties compared to the initialization. Moreover, the computational cost of overcomplete transform learning is lower than that of overcomplete dictionary learning. We also applied the adaptive overcomplete sparsifying transforms to image denoising, where they provide better performance over the synthesis K-SVD, while being faster. The overcomplete transforms also denoise better than square transforms. The promise of this overcomplete transform learning scheme in applications such as blind compressed sensing merits further study.

In Chapter 11, focusing again on the transform model for sparse representations, we presented a novel union of sparsifying transforms model. We showed that this model can also be interpreted as an overcomplete sparsifying transform model with an extra block cosparsity constraint (OCTOBOS) on the sparse code. The sparse coding in the proposed OCTOBOS model can be interpreted as a form of clustering. This makes the model potentially useful for classification tasks. We presented a novel problem formulation and algorithm for learning the proposed OCTOBOS transforms. Our algorithm involves simple closed-form solutions, and is thus computationally very efficient. Our theoretical analysis established global convergence (i.e., convergence from any initialization) of the algorithm to the set of partial minimizers of the non-convex learning problem. For natural images, our learning scheme gives rise to a union of well-conditioned transforms, and clustered patches or textures. It is also usually insensitive to initialization. The adapted model provides better sparsification errors and recovery PSNRs for images compared to learned single square transforms, and analytical transforms. In the application of image denoising, the proposed scheme typically provides better image reconstruction quality compared to adaptive (single) square transforms, and adaptive overcomplete synthesis dictionaries. These results suggest that the proposed OCTOBOS learning produces effective models adapted to the data. The usefulness of our OCTOBOS learning scheme in these applications merits detailed study and further evaluation. Likewise, other applications, e.g., inverse problems such as MRI and CT [97, 98], and classification merit further study. We also plan to perform detailed comparisons between the unstructured overcomplete transform learning scheme in Chapter 10 and the structured OCTOBOS scheme of Chapter 11 in future work.

Finally, I would like to mention that automating the parameter selection for the methods in this work remains an open problem that merits detailed investigation.

APPENDIX A

PROOFS FOR CHAPTER 2

A.1 Proof of Lemma 1

We now derive a lower bound for $Q = -\log\det W + c\,\|W\|_F^2$ that depends on the condition number κ of W. We work with the case $\det W > 0$. Denoting the singular values of W by $\beta_i, 1 \leq i \leq n$, we have $\|W\|_F^2 = \sum_{i=1}^n \beta_i^2$ and $-\log\det W = -\log(\prod_{i=1}^n \beta_i) = -\sum_{i=1}^n \log\beta_i$. Therefore,

$$Q = \sum_{i=1}^{n} \left(-\log\beta_i + c\beta_i^2\right) \tag{A.1}$$

We now bound the terms on the right hand side of (A.1). Since, $-\log\beta_i + c\beta_i^2$ is a strictly convex function of β_i for each i, we lower bound it using its minimum value (achieved when $\beta_i = \sqrt{\frac{1}{2c}}$) as follows.

$$-\log\beta_i + c\beta_i^2 \geq \frac{1}{2} + \frac{1}{2}\log(2c) \tag{A.2}$$

We substitute the above bound in equation (A.1) only for the indices $2 \leq i \leq n-1$ to get

$$Q \geq \frac{n-2}{2} + \frac{n-2}{2}\log(2c) - \log(\beta_1\beta_n) + c(\beta_1^2 + \beta_n^2) \tag{A.3}$$

We now express β_1 in terms of β_n in equation (A.3).

$$\beta_1 = \kappa\beta_n \tag{A.4}$$

Next, since, $-\log(\kappa\beta_n^2) + c\beta_n^2(1+\kappa^2)$ is a strictly convex function, we get the following lower bound using the minimum value (achieved when $\beta_n =$

$\sqrt{\frac{1}{c(1+\kappa^2)}}$) of the function.

$$-\log(\kappa\beta_n^2) + c\beta_n^2(1+\kappa^2) \geq 1 - \log\frac{\kappa}{c(1+\kappa^2)} \tag{A.5}$$

Upon substitution of the preceding bound in equation (A.3) and simplification, we get the following lower bound on Q.

$$Q \geq \frac{n}{2} + \frac{n}{2}\log(2c) - \log\frac{2\kappa}{1+\kappa^2} \qquad \blacksquare$$

which completes the proof.

A.2 Proof of Corollary 1

The inequality $Q \geq Q_0$ follows from Lemma 1, because $-\log\frac{2\kappa}{1+\kappa^2}$ has a minimum value of zero (achieved for $\kappa = 1$). For the result to hold with equality, we require that $\kappa = 1$. We also require (A.3) and (A.5) in the proof of Lemma 1 to hold with equality, which can happen if and only if

$$\beta_1 = \sqrt{\frac{\kappa^2}{c(1+\kappa^2)}}, \quad \beta_n = \sqrt{\frac{1}{c(1+\kappa^2)}} \tag{A.6}$$

$$\beta_i = \sqrt{\frac{1}{2c}}, \quad 2 \leq i \leq n-1$$

Now, using $\kappa = 1$ in (A.6), we get the required result. $\blacksquare$

APPENDIX B

PROOFS FOR CHAPTER 4

B.1 Proof of Proposition 3

First, in the sparse coding step of (P4.0), we solve (4.1) (or (4.2)) for $\hat{X}$ with a fixed W. Then, the $\hat{X}$ discussed in Section 4.3.1 does not depend on the weight λ, and it remains unaffected as $\lambda \to \infty$.

Next, in the transform update step, we solve for $\hat{W}$ in (4.4) with a fixed sparse code X. The transform update solution (4.5) does depend on the weight λ. For a particular λ, let us choose the matrix L_λ (indexed by λ) as the positive-definite square root $\left(YY^T + 0.5\lambda I\right)^{1/2}$. By Proposition 2, the closed-form formula (4.5) is invariant to the specific choice of this matrix. Let us define matrix M_λ as

$$M_\lambda \triangleq \sqrt{0.5\lambda}\, L_\lambda^{-1} Y X^T = \left[(2/\lambda) YY^T + I\right]^{-\frac{1}{2}} YX^T \tag{B.1}$$

and its full SVD as $Q_\lambda \tilde{\Sigma}_\lambda R_\lambda^T$. As $\lambda \to \infty$, by (B.1), $M_\lambda = Q_\lambda \tilde{\Sigma}_\lambda R_\lambda^T$ converges to $M = YX^T$, and it can be shown (see Appendix B.2) that the accumulation points of $\{Q_\lambda\}$ and $\{R_\lambda\}$ (considering the sequences indexed by λ, and letting $\lambda \to \infty$) belong to the set of left and right singular matrices of YX^T, respectively. Moreover, as $\lambda \to \infty$, the matrix $\tilde{\Sigma}_\lambda$ converges to a non-negative $n \times n$ diagonal matrix, which is the matrix of singular values of YX^T.

On the other hand, using (B.1) and the SVD of M_λ, (4.5) can be rewritten as follows

$$\hat{W}_\lambda = R_\lambda \left[\frac{\tilde{\Sigma}_\lambda}{\lambda} + \left(\frac{\tilde{\Sigma}_\lambda^2}{\lambda^2} + I \right)^{\frac{1}{2}} \right] Q_\lambda^T \left(\frac{YY^T}{0.5\lambda} + I \right)^{-\frac{1}{2}}$$

In the limit of $\lambda \to \infty$, using the aforementioned arguments on the limiting behavior of $\{Q_\lambda\}$, $\{\tilde{\Sigma}_\lambda\}$, and $\{R_\lambda\}$, the above update formula becomes (or, when YX^T has some degenerate singular values, the accumulation point(s) of the above formula assume the following form)

$$\hat{W} = \hat{R}\hat{Q}^T \tag{B.2}$$

where $\hat{Q}$ and $\hat{R}$ above are the full left and right singular matrices of YX^T, respectively. (Note that for $\xi \neq 0.5$, the right hand side of (B.2) is simply scaled by the constant $1/\sqrt{2\xi}$.) It is clear that the updated transform in (B.2) above is orthonormal.

Importantly, as $\lambda \to \infty$ (with $\xi = 0.5$), the sparse coding and transform update solutions in (P4.0) coincide with the corresponding solutions obtained by employing alternating minimization on the orthonormal transform learning Problem (4.14). Specifically, the sparse coding step for Problem (4.14) involves the same aforementioned Problem (4.1). Furthermore, using the condition $W^TW = I$, it is easy to show that the minimization problem in the transform update step of Problem (4.14) simplifies to the form in (4.15). Problem (4.15) is of the form of the well-known orthogonal Procrustes problem [199]. Therefore, denoting the full SVD of YX^T by $U\Sigma V^T$, the optimal solution in Problem (4.15) is given exactly as $\hat{W} = VU^T$. It is now clear that the solution for W in the orthonormal transform update Problem (4.15) is identical to the limit shown in (B.2).

Lastly, the solution to Problem (4.15) is unique if and only if the singular values of YX^T are non-zero. The reasoning for the latter statement is similar to that provided in the proof of Proposition 2 (in Section 4.3.1) for the uniqueness of the transform update solution for Problem (P4.0). ■

B.2 Limit of a Sequence of Singular Value Decompositions

Lemma 15. *Consider a sequence $\{M_k\}$ with $M_k \in \mathbb{R}^{n\times n}$, that converges to M. For each k, let $Q_k\Sigma_kR_k^T$ denote a full SVD of M_k. Then, every accumulation point* [1] *(Q, Σ, R) of the sequence $\{Q_k, \Sigma_k, R_k\}$ is such that $Q\Sigma R^T$ is a*

[1]Non-uniqueness of the accumulation point may arise due to the fact that the left and right singular vectors in the singular value decomposition (of M_k, M) are non-unique.

full SVD of M. In particular, $\{\Sigma_k\}$ converges to Σ, the $n \times n$ singular value matrix of M.

Proof: Consider a convergent subsequence $\{Q_{q_k}, \Sigma_{q_k}, R_{q_k}\}$ of the sequence $\{Q_k, \Sigma_k, R_k\}$, that converges to the accumulation point (Q, Σ, R). It follows that

$$\lim_{k\to\infty} M_{q_k} = \lim_{k\to\infty} Q_{q_k} \Sigma_{q_k} R_{q_k}^T = Q\Sigma R^T \tag{B.3}$$

Obviously, the subsequence $\{M_{q_k}\}$ converges to the same limit M as the (original) sequence $\{M_k\}$. Therefore, we have

$$M = Q\Sigma R^T \tag{B.4}$$

By the continuity of inner products, the limit of a sequence of orthonormal matrices is orthonormal. Therefore the limits Q and R of the orthonormal subsequences $\{Q_{q_k}\}$ and $\{R_{q_k}\}$ are themselves orthonormal. Moreover, Σ, being the limit of a sequence $\{\Sigma_{q_k}\}$ of non-negative diagonal matrices (each with decreasing diagonal entries), is also a non-negative diagonal (the limit maintains the decreasing ordering of the diagonal elements) matrix. By these properties and (B.4), it is clear that $Q\Sigma R^T$ is a full SVD of M. The preceding arguments also indicate that the accumulation point of $\{\Sigma_k\}$ is unique, i.e., Σ. In other words, $\{\Sigma_k\}$ converges to Σ, the singular value matrix of M. $\blacksquare$

B.3 Main Convergence Proof

Here, we present the proof of convergence for our alternating algorithm for (P4.0), i.e., proof of Theorem 1. The proof for Theorem 2 is very similar to that for Theorem 1. The only difference is that the non-negative barrier function $\psi(X)$ and the operator $H_s(\cdot)$ (in the proof of Theorem 1) are replaced by the non-negative penalty $\sum_{i=1}^{N} \eta_i^2 \|X_i\|_0$ and the operator $\hat{H}_\eta^1(\cdot)$, respectively. Hence, for brevity, we only provide a sketch of the proof of Theorem 2.

We will use the operation $\tilde{H}_s(b)$ here to denote the *set* of all optimal projections of $b \in \mathbb{R}^n$ onto the s-ℓ_0 ball, i.e., $\tilde{H}_s(b)$ is the set of all minimizers in the following problem.

$$\tilde{H}_s(b) = \underset{x \,:\, \|x\|_0 \leq s}{\arg\min} \; \|x - b\|_2^2 \tag{B.5}$$

Similarly, in the case of Theorem 2, the operation $\hat{H}_\eta(b)$ is defined as a mapping of a vector b to a set as

$$\left(\hat{H}_\eta(b)\right)_j = \begin{cases} 0 & , \ |b_j| < \eta \\ \{b_j, \, 0\} & , \ |b_j| = \eta \\ b_j & , \ |b_j| > \eta \end{cases} \tag{B.6}$$

The set $\hat{H}_\eta(b)$ is in fact, the set of all optimal solutions to (4.2), when Y is replaced by the vector b, and $\eta_1 = \eta$.

Theorem 1 is now proved by proving the following properties one-by-one.

(i) Convergence of the objective in Algorithm A1.

(ii) Existence of an accumulation point for the iterate sequence generated by Algorithm A1.

(iii) All the accumulation points of the iterate sequence are equivalent in terms of their objective value.

(iv) Every accumulation point of the iterate sequence is a fixed point of the algorithm.

(v) Every fixed point of the algorithm is a local minimizer of $g(W, X)$ in the sense of (4.16).

The following shows the convergence of the objective.

Lemma 16. *Let $\left\{W^k, X^k\right\}$ denote the iterate sequence generated by Algorithm A1 with data Y and initial (W^0, X^0). Then, the sequence of objective function values $\left\{g(W^k, X^k)\right\}$ is monotone decreasing, and converges to a finite value $g^* = g^*\left(W^0, X^0\right)$.*

Proof: In the transform update step, we obtain a global minimizer with respect to W in the form of the closed-form analytical solution (4.5). Thus, the objective can only decrease in this step, i.e., $g(W^{k+1}, X^k) \leq g(W^k, X^k)$.

In the sparse coding step too, we obtain an exact solution for X with fixed W as $\hat{X}_i = H_s(WY_i)\ \forall\, i$. Thus, $g(W^{k+1}, X^{k+1}) \leq g(W^{k+1}, X^k)$. Combining the results for the two steps, we have $g(W^{k+1}, X^{k+1}) \leq g(W^k, X^k)$ for any k.

Now, in Section 4.2 of Chapter 4, we stated an explicit lower bound for the function $g(W, X) - \psi(X)$. Since $\psi(X) \geq 0$, we therefore have that the function $g(W, X)$ is also lower bounded. Since the sequence of objective function values $\left\{g(W^k, X^k)\right\}$ is monotone decreasing and lower bounded, it must converge. $\blacksquare$

Lemma 17. *The iterate sequence $\left\{W^k, X^k\right\}$ generated by Algorithm A1 is bounded, and it has at least one accumulation point.*

Proof: The existence of a convergent subsequence for a bounded sequence is a standard result. Therefore, a bounded sequence has at least one accumulation point. We now prove the boundedness of the iterates. Let us denote $g(W^k, X^k)$ as g^k for simplicity. We then have the boundedness of $\left\{W^k\right\}$ as follows. First, since g^k is the sum of $v(W^k)$, and the non-negative sparsification error and $\psi(X^k)$ terms, we have that

$$v(W^k) \leq g^k \leq g^0 \tag{B.7}$$

where the second inequality above follows from Lemma 16. Denoting the singular values of W^k by β_i $(1 \leq i \leq n)$, we have that $v(W^k) = \sum_{i=1}^{n}(\xi\beta_i^2 - \log\, \beta_i)$. The function $\sum_{i=1}^{n}(\xi\beta_i^2 - \log\, \beta_i)$, as a function of the singular values $\{\beta_i\}_{i=1}^{n}$ (all positive) is strictly convex, and it has bounded lower level sets. (Note that the level sets of a function $f : A \subset \mathbb{R}^n \mapsto \mathbb{R}$ (where A is unbounded) are bounded if $\lim_{k\to\infty} f(x^k) = +\infty$ whenever $\left\{x^k\right\} \subset A$ and $\lim_{k\to\infty} \left\|x^k\right\| = \infty$.) This fact, together with (B.7) implies that $\left\|W^k\right\|_F = \sqrt{\sum_{i=1}^{n}\beta_i^2} \leq c_0$ for a constant c_0, that depends on g^0. The same bound (c_0) works for any k.

We also have the following inequalities for sequence $\left\{X^k\right\}$.

$$\left\|X^k\right\|_F - \left\|W^kY\right\|_F \leq \left\|W^kY - X^k\right\|_F \leq \sqrt{g^k - v_0}$$

The first inequality follows from the triangle inequality and the second inequality follows from the fact that g^k is the sum of the sparsification error and $v(W^k)$ terms (since $\psi(X^k) = 0$), and $v(W^k) \geq v_0$ (v_0 defined in Section

4.2) [9]. By Lemma 16, $\sqrt{g^k - v_0} \leq \sqrt{g^0 - v_0}$. Denoting $\sqrt{g^0 - v_0}$ by c_1, we have

$$\left\|X^k\right\|_F \leq c_1 + \left\|W^k Y\right\|_F \leq c_1 + \sigma_1 \left\|W^k\right\|_F \tag{B.8}$$

where σ_1 is the largest singular value of the matrix Y. The boundedness of $\left\{X^k\right\}$ then follows from the previously established fact that $\left\|W^k\right\|_F \leq c_0$. ■

We now prove some important properties (Lemmas 18, 19, and 20) satisfied by any accumulation point of the iterate sequence $\left\{W^k, X^k\right\}$ in our algorithm.

Lemma 18. *Any accumulation point* (W^*, X^*) *of the iterate sequence* $\left\{W^k, X^k\right\}$ *generated by Algorithm A1 satisfies*

$$X_i^* \in \tilde{H}_s(W^* Y_i) \;\; \forall \; i \tag{B.9}$$

Proof: Let $\{W^{q_k}, X^{q_k}\}$ be a subsequence of the iterate sequence converging to the accumulation point (W^*, X^*). It is obvious that W^* is non-singular. Otherwise, the objective cannot be monotone decreasing over $\{W^{q_k}, X^{q_k}\}$.

We now have that for each (column) i $(1 \leq i \leq N)$,

$$X_i^* = \lim_{k \to \infty} X_i^{q_k} = \lim_{k \to \infty} H_s(W^{q_k} Y_i) \in \tilde{H}_s(W^* Y_i) \tag{B.10}$$

where we have used the fact that when a vector sequence $\left\{\alpha^k\right\}$ converges to α^*, then the accumulation point of the sequence $\left\{H_s(\alpha^k)\right\}$ lies in $\tilde{H}_s(\alpha^*)$ [2] (see proof in Appendix B.4). ■

Lemma 19. *All the accumulation points of the iterate sequence* $\left\{W^k, X^k\right\}$ *generated by Algorithm A1 with initial* (W^0, X^0) *correspond to the same objective value. Thus, they are equivalent in that sense.*

Proof: Let $\{W^{q_k}, X^{q_k}\}$ be a subsequence of the iterate sequence converging to an accumulation point (W^*, X^*). Define a function $g'(W, X) = g(W, X) - \psi(X)$. Then, for any non-singular W, $g'(W, X)$ is continuous in its arguments. Moreover, for the subsequence $\{W^{q_k}, X^{q_k}\}$ and its accumulation

[2]Since the mapping $H_s(\cdot)$ is discontinuous, the sequence $\left\{H_s(\alpha^k)\right\}$ need not converge to $H_s(\alpha^*)$, even though $\left\{\alpha^k\right\}$ converges to α^*.

point ($X_i^* \in \tilde{H}_s(W^* Y_i)\ \forall\ i$ by Lemma 18), the barrier function $\psi(X) = 0$. Therefore,

$$\begin{aligned}\lim_{k\to\infty} g(W^{q_k}, X^{q_k}) &= \lim_{k\to\infty} g'(W^{q_k}, X^{q_k}) + \lim_{k\to\infty} \psi(X^{q_k}) \\ &= g'(W^*, X^*) + 0 = g(W^*, X^*) \qquad \text{(B.11)}\end{aligned}$$

where we have used the continuity of g' at (W^*, X^*) (since W^* is non-singular). Now, since, by Lemma 16, the objective converges for Algorithm A1, we have that $\lim_{k\to\infty} g(W^{q_k}, X^{q_k}) = \lim_{k\to\infty} g(W^k, X^k) = g^*$. Combining with (B.11), we have

$$g^* = g(W^*, X^*) \qquad \text{(B.12)}$$

Equation (B.12) indicates that any accumulation point (W^*, X^*) of the iterate sequence $\{W^k, X^k\}$ satisfies $g(W^*, X^*) = g^*$, with g^* being the limit of $\{g(W^k, X^k)\}$. ■

Lemma 20. *Any accumulation point (W^*, X^*) of the iterate sequence $\{W^k, X^k\}$ generated by Algorithm A1 satisfies*

$$W^* \in \arg\min_{W} \|WY - X^*\|_F^2 + \lambda\xi \|W\|_F^2 - \lambda \log |\det W| \qquad \text{(B.13)}$$

Proof: Let $\{W^{q_k}, X^{q_k}\}$ be a subsequence of the iterate sequence converging to the accumulation point (W^*, X^*). We then have (due to linearity) that

$$\lim_{k\to\infty} L^{-1} Y \left(X^{q_k}\right)^T = L^{-1} Y \left(X^*\right)^T \qquad \text{(B.14)}$$

Let $Q^{q_k} \Sigma^{q_k} \left(R^{q_k}\right)^T$ denote the full singular value decomposition of $L^{-1} Y \left(X^{q_k}\right)^T$. Then, by Lemma 15 of Appendix B.2, we have that every accumulation point (Q^*, Σ^*, R^*) of the sequence $\{Q^{q_k}, \Sigma^{q_k}, R^{q_k}\}$ is such that $Q^* \Sigma^* \left(R^*\right)^T$ is a full SVD of $L^{-1} Y \left(X^*\right)^T$, i.e.,

$$Q^* \Sigma^* \left(R^*\right)^T = L^{-1} Y \left(X^*\right)^T \qquad \text{(B.15)}$$

In particular, $\{\Sigma^{q_k}\}$ converges to Σ^*, the full singular value matrix of $L^{-1} Y \left(X^*\right)^T$. Now, for a convergent subsequence of $\{Q^{q_k}, \Sigma^{q_k}, R^{q_k}\}$ (with limit (Q^*, Σ^*, R^*)) , using the closed-form formula (4.5), we have

$$\begin{aligned} W^{**} &\triangleq \lim_{k\to\infty} W^{q_{n_k}+1} \\ &= \lim_{k\to\infty} \frac{R^{q_{n_k}}}{2}\left(\Sigma^{q_{n_k}} + \left((\Sigma^{q_{n_k}})^2 + 2\lambda I\right)^{\frac{1}{2}}\right)\left(Q^{q_{n_k}}\right)^T L^{-1} \\ &= \frac{R^*}{2}\left(\Sigma^* + \left((\Sigma^*)^2 + 2\lambda I\right)^{\frac{1}{2}}\right)\left(Q^*\right)^T L^{-1} \end{aligned} \tag{B.16}$$

where the last equality in (B.16) follows from the continuity of the square root function, and the fact that $\lambda > 0$. Equations (B.15), (B.16), and (4.5) imply that

$$W^{**} \in \arg\min_W \|WY - X^*\|_F^2 + \lambda\xi \|W\|_F^2 - \lambda \log |\det W| \tag{B.17}$$

Now, applying the same arguments used in (B.11) and (B.12) to the sequence $\{W^{q_{n_k}+1}, X^{q_{n_k}}\}$, we get that $g^* = g(W^{**}, X^*)$. Combining with (B.12), we get $g(W^{**}, X^*) = g(W^*, X^*)$, i.e., for a fixed sparse code X^*, W^* achieves the same value of the objective as W^{**}. This result together with (B.17) proves the required result (B.13). ■

Next, we use Lemmas 18 and 20 to show that any accumulation point of the iterate sequence in Algorithm A1 is a fixed point of the algorithm.

Lemma 21. *Any accumulation point of the iterate sequence* $\left\{W^k, X^k\right\}$ *generated by Algorithm A1 is a fixed point of the algorithm.*

Proof: Let $\{W^{q_k}, X^{q_k}\}$ be a subsequence of the iterate sequence converging to some accumulation point (W^*, X^*). Lemmas 18 and 20 then imply that

$$X^* \in \arg\min_X g(W^*, X) \tag{B.18}$$

$$W^* \in \arg\min_W g(W, X^*) \tag{B.19}$$

In order to deal with any non-uniqueness of solutions above, we assume for our algorithm that if a certain iterate W^{k+1} satisfies $g(W^{k+1}, X^k) = g(W^k, X^k)$, then we equivalently set $W^{k+1} = W^k$. Similarly, if $W^{k+1} = W^k$ holds, then we set $X^{k+1} = X^k$ [3]. Under the preceding assumptions, equations (B.19) and (B.18) imply that if we feed (W^*, X^*) into our alternating

[3]This rule is trivially true, except when applied to say an accumulation point X^* (i.e., replace X^k by X^* in the rule) such that $X_i^* \in \tilde{H}_s(W^*Y_i)\ \forall\ i$, but $X_i^* \neq H_s(W^*Y_i)$ for some i.

algorithm (as initial estimates), the algorithm stays at (W^*, X^*). In other words, the accumulation point (W^*, X^*) is a fixed point of the algorithm. $\blacksquare$

Finally, the following Lemma 23 shows that any accumulation point (i.e., fixed point by Lemma 21) of the iterates is a local minimizer. Since the accumulation points are equivalent in terms of their cost, Lemma 23 implies that they are equally good local minimizers.

We will also need the following simple lemma for the proof of Lemma 23.

Lemma 22. *The function $f(G) = tr(G) - \log|\det(I+G)|$ for $G \in \mathbb{R}^{n\times n}$, has a strict local minimum at $G = 0$, i.e., there exists an $\epsilon > 0$ such that for $\|G\|_F \leq \epsilon$, we have $f(G) \geq f(0) = 0$, with equality attained only at $G = 0$.*

Proof: The gradient of $f(G)$ (when it exists) is given [9] as

$$\nabla_G f(G) = I - (I+G)^{-T} \tag{B.20}$$

It is clear that $G = 0$ produces a zero (matrix) value for the gradient. Thus, $G = 0$ is a stationary point of $f(G)$. The Hessian of $f(G)$ can also be derived [200] as $H = (I+G)^{-T} \otimes (I+G)^{-1}$, where "$\otimes$" denotes the Kronecker product. The Hessian is I_{n^2} at $G = 0$. Since this Hessian is positive definite, it means that $G = 0$ is a strict local minimizer of $f(G)$. The rest of the lemma is trivial. $\blacksquare$

Lemma 23. *Every fixed point (W, X) of our Algorithm A1 is a minimizer of the objective $g(W, X)$ of Problem (P4.0), in the sense of* (4.16) *for sufficiently small dW, and ΔX in the union of the regions R1 and R2 in Theorem 1. Furthermore, if $\|WY_i\|_0 \leq s\,\forall\, i$, then ΔX can be arbitrary.*

Proof: It is obvious that W is a global minimizer of the transform update problem (4.4) for fixed sparse code X, and it thus provides a gradient value of 0 for the objective of (4.4). Thus, we have (using gradient expressions from [9])

$$2WYY^T - 2XY^T + 2\lambda\xi W - \lambda W^{-T} = 0 \tag{B.21}$$

Additionally, we also have the following optimal property for the sparse code.

$$X_i \in \tilde{H}_s(WY_i)\ \ \forall\, i \tag{B.22}$$

Now, given such a fixed point (W, X), we consider perturbations $dW \in \mathbb{R}^{n\times n}$, and $\Delta X \in \mathbb{R}^{n\times N}$. We are interested in the relationship between $g(W + dW, X + \Delta X)$ and $g(W, X)$. It suffices to consider *sparsity preserving* ΔX, that is ΔX such that $X+\Delta X$ has columns that have sparsity $\leq s$. Otherwise the barrier function $\psi(X+\Delta X) = +\infty$, and $g(W+dW, X+\Delta X) > g(W, X)$ trivially. Therefore, in the rest of the proof, we only consider sparsity preserving ΔX.

For sparsity preserving $\Delta X \in \mathbb{R}^{n\times N}$, we have

$$\begin{aligned} g(W + dW, X + \Delta X) &= \|WY - X + (dW)Y - \Delta X\|_F^2 \\ &+ \lambda\xi \, \|W + dW\|_F^2 - \lambda \, \log \, |\det{(W + dW)}| \end{aligned} \tag{B.23}$$

Expanding the two Frobenius norm terms above using the trace inner product $\langle Q, R\rangle \triangleq \operatorname{tr}(QR^T)$, and dropping the non-negative terms $\|(dW)Y - \Delta X\|_F^2$ and $\lambda\xi \, \|dW\|_F^2$, we obtain

$$\begin{aligned} g(W + dW, X + \Delta X) \geq\ & \|WY - X\|_F^2 + \lambda\xi \, \|W\|_F^2 \\ & + 2\,\langle WY - X, (dW)Y - \Delta X\rangle + 2\lambda\xi \, \langle W, dW\rangle \\ & - \lambda \, \log \, |\det{(W + dW)}| \end{aligned} \tag{B.24}$$

Using (B.21) and the identity $\log\,|\det{(W + dW)}| = \log\,|\det W| + \log\,|\det{(I + W^{-1}dW)}|$, equation (B.24) simplifies to

$$\begin{aligned} g(W + dW, X + \Delta X) &\geq g(W, X) + \lambda \left\langle W^{-T}, dW\right\rangle \\ &- 2\,\langle WY - X, \Delta X\rangle - \lambda \, \log \, \left|\det{(I + W^{-1}dW)}\right| \end{aligned} \tag{B.25}$$

Define $G \triangleq W^{-1}dW$. Then, the terms $\left\langle W^{-T}, dW\right\rangle - \log\,|\det{(I + W^{-1}dW)}|$ (appearing in (B.25) with a scaling λ) coincide with the function $f(G)$ in Lemma 22. Therefore, by Lemma 22, we have that there exists an $\epsilon > 0$ such that for $\|W^{-1}dW\|_F \leq \epsilon$, we have $\left\langle W^{-T}, dW\right\rangle - \log\,|\det{(I + W^{-1}dW)}| \geq 0$, with equality attained here only at $dW = 0$. Since $\|W^{-1}dW\|_F \leq \|dW\|_F / \sigma_n$, where σ_n is the smallest singular value of W, we have that an alternative sufficient condition (for the aforementioned positivity of $f(W^{-1}dW)$) is $\|dW\|_F \leq \epsilon\sigma_n$. Assuming that dW lies in this neighborhood, equation (B.25) becomes

$$g(W + dW, X + \Delta X) \geq g(W, X) - 2 \langle WY - X, \Delta X \rangle \tag{B.26}$$

Thus, we have the optimality condition $g(W + dW, X + \Delta X) \geq g(W, X)$ for any $dW \in \mathbb{R}^{n \times n}$ satisfying $\|dW\|_F \leq \epsilon \sigma_n$ (ϵ from Lemma 22), and for any $\Delta X \in \mathbb{R}^{n \times N}$ satisfying $\langle WY - X, \Delta X \rangle \leq 0$. This result defines Region R1.

We can also define a simple local region $\mathrm{R2} \subseteq \mathbb{R}^{n \times N}$, such that any sparsity preserving ΔX in the region results in $\langle WY - X, \Delta X \rangle = 0$. Then, by (B.26), $g(W + dW, X + \Delta X) \geq g(W, X)$ holds for $\Delta X \in \mathrm{R2}$. As we now show, the region R2 includes all $\Delta X \in \mathbb{R}^{n \times N}$ satisfying $\|\Delta X\|_\infty < \min_i \{\phi_s(WY_i) : \|WY_i\|_0 > s\}$. In the definition of R2, we need only consider the columns of WY with sparsity $> s$. To see why, consider the set $\mathcal{A} \triangleq \{i : \|WY_i\|_0 > s\}$, and its complement $\mathcal{A}^c = \{1, ..., N\} \setminus \mathcal{A}$. Then, we have

$$\begin{aligned} \langle WY - X, \Delta X \rangle &= \sum_{i \in \mathcal{A} \cup \mathcal{A}^c} \Delta X_i^T (WY_i - X_i) \\ &= \sum_{i \in \mathcal{A}} \Delta X_i^T (WY_i - X_i) \end{aligned} \tag{B.27}$$

where we used the fact that $WY_i - X_i = 0$, $\forall\ i \in \mathcal{A}^c$. It is now clear that $\langle WY - X, \Delta X \rangle$ is unaffected by the columns of WY with sparsity $\leq s$. Therefore, these columns do not appear in the definition of R2. Moreover, if $\mathcal{A} = \emptyset$, then $\langle WY - X, \Delta X \rangle = 0$ for arbitrary $\Delta X \in \mathbb{R}^{n \times N}$, and thus, $g(W + dW, X + \Delta X) \geq g(W, X)$ holds (by (B.26)) for arbitrary ΔX. This proves the last statement of the Lemma.

Otherwise, assume $\mathcal{A} \neq \emptyset$, $\Delta X \in \mathrm{R2}$, and recall from (B.22) that $X_i \in \tilde{H}_s(WY_i)\ \forall\ i$. It follows by the definition of R2, that for $i \in \mathcal{A}$, any $X_i + \Delta X_i$ with sparsity $\leq s$ will have the same sparsity pattern (non-zero locations) as X_i, i.e., the corresponding ΔX_i does not have non-zeros outside the support of X_i. Now, since $X_i \in \tilde{H}_s(WY_i)$, $WY_i - X_i$ is zero on the support of X_i, and thus, $\Delta X_i^T (WY_i - X_i) = 0$ for all $i \in \mathcal{A}$. Therefore, by (B.27), $\langle WY - X, \Delta X \rangle = 0$, for any sparsity-preserving ΔX in R2. ■

Note that the proof of Theorem 2 also requires Lemma 23, but with the objective $g(W, X)$ replaced by $u(W, X)$. Appendix B.5 briefly discusses how the proof of the Lemma 23 is modified for the case of Theorem 2.

B.4 Limit of a Thresholded Sequence

Lemma 24. *Consider a bounded vector sequence $\{\alpha^k\}$ with $\alpha^k \in \mathbb{R}^n$, that converges to α^*. Then, every accumulation point of $\{H_s(\alpha^k)\}$ belongs to the set $\tilde{H}_s(\alpha^*)$.*

Proof: If $\alpha^* = 0$, then it is obvious that $\{H_s(\alpha^k)\}$ converges to $\tilde{H}_s(\alpha^*) = 0$. Therefore, we now only consider the case $\alpha^* \neq 0$.

First, let us assume that $\tilde{H}_s(\alpha^*)$ (the set of optimal projections of α^* onto the s-ℓ_0 ball) is a singleton and $\phi_s(\alpha^*) > 0$, so that $\phi_s(\alpha^*) - \phi_{s+1}(\alpha^*) > 0$. Then, for sufficiently large k ($k \geq k_0$), we will have $\left\|\alpha^k - \alpha^*\right\|_\infty < (\phi_s(\alpha^*) - \phi_{s+1}(\alpha^*))/2$, and then, $H_s(\alpha^k)$ has the same support set (non-zero locations) Γ as $H_s(\alpha^*) = \tilde{H}_s(\alpha^*)$. As $k \to \infty$, since $\left\|\alpha^k_\Gamma - \alpha^*_\Gamma\right\|_2 \to 0$ (where the subscript Γ indicates that only the elements of the vector corresponding to the support Γ are considered), we have that $\left\|H_s(\alpha^k) - H_s(\alpha^*)\right\|_2 \to 0$. Thus, the sequence $\{H_s(\alpha^k)\}$ converges to $H_s(\alpha^*)$ in this case.

Next, when $\tilde{H}_s(\alpha^*)$ is a singleton, but $\phi_s(\alpha^*) = 0$ (and $\alpha^* \neq 0$), let γ be the magnitude of the non-zero element of α^* of smallest magnitude. Then, for sufficiently large k ($k \geq k_1$), we will have $\left\|\alpha^k - \alpha^*\right\|_\infty < \gamma/2$, and then, the support of $H_s(\alpha^*) = \tilde{H}_s(\alpha^*)$ is contained in the support of $H_s(\alpha^k)$. Therefore, for $k \geq k_1$, we have

$$\left\|H_s(\alpha^k) - H_s(\alpha^*)\right\|_2 = \sqrt{\left\|\alpha^k_{\Gamma_1} - \alpha^*_{\Gamma_1}\right\|_2^2 + \left\|\alpha^k_{\Gamma_2}\right\|_2^2} \tag{B.28}$$

where Γ_1 is the support set of $H_s(\alpha^*)$, and Γ_2 (depends on k) is the support set of $H_s(\alpha^k)$ excluding Γ_1. (Note that α^* and $H_s(\alpha^*)$ are zero on Γ_2.) As $k \to \infty$, since $\alpha^k \to \alpha^*$, we have that $\left\|\alpha^k_{\Gamma_1} - \alpha^*_{\Gamma_1}\right\|_2 \to 0$ and $\left\|\alpha^k_{\Gamma_2}\right\|_2 \to 0$. Combining this with (B.28), we then have that the sequence $\{H_s(\alpha^k)\}$ converges to $H_s(\alpha^*)$ in this case too.

Finally, when $\tilde{H}_s(\alpha^*)$ is not a singleton (there are ties), it is easy to show that for sufficiently large k ($k \geq k_2$), the support of $H_s(\alpha^k)$ for each k coincides with the support of one of the optimal codes in $\tilde{H}_s(\alpha^*)$. In this case, as $k \to \infty$ (or, as $\alpha^k \to \alpha^*$), the distance between $H_s(\alpha^k)$ and the set $\tilde{H}_s(\alpha^*)$ converges to 0. Therefore, the accumulation point(s) of $\{H_s(\alpha^k)\}$ in this case, all belong to the set $\tilde{H}_s(\alpha^*)$. $\blacksquare$

Specifically, in the case of equation (B.10), Lemma 24 implies that $X_i^* \in \tilde{H}_s(W^* Y_i)$.

B.5 Modifications to Proof of Lemma 23 for Theorem 2

The (unconstrained) objective $u(W, X)$ here does not have the barrier function $\psi(X)$, but instead the penalty $\sum_{i=1}^{N} \eta_i^2 \|X_i\|_0$. Let us consider a fixed point (W, X) of the alternating Algorithm A2 that minimizes this objective. For a perturbation $\Delta X \in \mathbb{R}^{n \times N}$ satisfying $\|\Delta X\|_\infty < \min_i \{\eta_i/2\}$, it is easy to see (since X satisfies $X_i \in \hat{H}_{\eta_i}(WY_i)\ \forall\, i$) that

$$\sum_{i=1}^{N} \eta_i^2 \|X_i + \Delta X_i\|_0 = \sum_{i=1}^{N} \eta_i^2 \|X_i\|_0 + \sum_{i=1}^{N} \eta_i^2 \|\Delta X_i^c\|_0 \tag{B.29}$$

where $\Delta X_i^c \in \mathbb{R}^n$ is zero on the support (non-zero locations) of X_i, and matches ΔX_i on the complement of the support of X_i.

Now, upon repeating the steps in the proof of Lemma 23 for the case of Theorem 2, we arrive at the following counterpart of equation (B.26).

$$\begin{aligned} u(W + dW, X + \Delta X) \geq & u(W, X) - 2\langle WY - X, \Delta X\rangle \\ & + \sum_{i=1}^{N} \eta_i^2 \|\Delta X_i^c\|_0 \end{aligned} \tag{B.30}$$

The term $-2\langle WY - X, \Delta X\rangle + \sum_{i=1}^{N} \eta_i^2 \|\Delta X_i^c\|_0$ above can be easily shown to be ≥ 0 for ΔX satisfying $\|\Delta X\|_\infty < \min_i \{\eta_i/2\}$. $\blacksquare$

APPENDIX C

PROOFS FOR CHAPTER 5

C.1 Proof of Theorem 3

In this proof, we let $\tilde{H}_s(Z)$ denote the *set* of all optimal projections of $Z \in \mathbb{C}^{n\times N}$ onto the s-ℓ_0 ball $\left\{B \in \mathbb{C}^{n\times N} : \|B\|_0 \leq s\right\}$, i.e., $\tilde{H}_s(Z)$ is the set of minimizers for the following problem.

$$\tilde{H}_s(Z) = \underset{B\,:\,\|B\|_0 \leq s}{\arg\min}\ \|B - Z\|_F^2 \tag{C.1}$$

We let $\{W^t, B^t, x^t\}$ denote the iterate sequence generated by Algorithm A1 with measurements $y \in \mathbb{C}^m$ and initial (W^0, B^0, x^0). We assume that the initial (W^0, B^0, x^0) is such that $g\left(W^0, B^0, x^0\right)$ is finite. The various results in Theorem 3 are now proved in the following order.

(i) Convergence of the objective in Algorithm A1.

(ii) Existence of an accumulation point for the iterate sequence generated by Algorithm A1.

(iii) All the accumulation points of the iterate sequence are equivalent in terms of their objective value.

(iv) Every accumulation point of the iterates is a critical point of the objective $g\left(W, B, x\right)$ satisfying (5.27), (5.28), and (5.29).

(v) The difference between successive image iterates $\|x^t - x^{t-1}\|_2$, converges to zero.

(vi) Every accumulation point of the iterates is a local minimizer of $g\left(W, B, x\right)$ with respect to (B, x) or (W, B).

The following two Lemmas establish the convergence of the objective, and the boundedness of the iterate sequence.

Lemma 25. *Let $\{W^t, B^t, x^t\}$ denote the iterate sequence generated by Algorithm A1 with input $y \in \mathbb{C}^m$ and initial (W^0, B^0, x^0). Then, the sequence of objective function values $\{g(W^t, B^t, x^t)\}$ is monotone decreasing, and converges to a finite value $g^* = g^*(W^0, B^0, x^0)$.*

Proof. : Algorithm A1 first alternates between the transform update and sparse coding steps (Step 2 in Fig. 5.1), with fixed image x. In the transform update step, we obtain a global minimizer (i.e., (5.11)) with respect to W for Problem (5.9). In the sparse coding step too, we obtain an exact solution for B in Problem (5.5) as $\hat{B} = H_s(Z)$. Therefore, the objective function can only decrease when we alternate between the transform update and sparse coding steps (similar to the case in [96]). Thus, we have $g(W^{t+1}, B^{t+1}, x^t) \leq g(W^t, B^t, x^t)$.

In the image update step of Algorithm A1 (Step 4 in Fig. 5.1), we obtain an exact solution to the constrained least squares problem (5.14). Therefore, the objective in this step satisfies $g(W^{t+1}, B^{t+1}, x^{t+1}) \leq g(W^{t+1}, B^{t+1}, x^t)$. Based on the preceding arguments, we have $g(W^{t+1}, B^{t+1}, x^{t+1}) \leq g(W^t, B^t, x^t)$, for every t. Now, every term, except $\lambda Q(W)$, in the objective (5.24) is trivially non-negative. Furthermore, the $Q(W)$ regularizer is bounded as $Q(W) \geq \frac{n}{2}$ (cf. [9]). Therefore, the objective $g(W, B, x) > 0$. Since the sequence $\{g(W^t, B^t, x^t)\}$ is monotone decreasing and lower bounded, it converges. $\square$

Lemma 26. *The iterate sequence $\{W^t, B^t, x^t\}$ generated by Algorithm A1 is bounded, and it has at least one accumulation point.*

Proof. : The existence of a convergent subsequence (and hence, an accumulation point) for a bounded sequence is a standard result. Therefore, we only prove the boundedness of the iterate sequence.

Since $\|x^t\|_2 \leq C \ \forall\ t$ trivially, we have that the sequence $\{x^t\}$ is bounded. We now show the boundedness of $\{W^t\}$. Let us denote the objective $g(W^t, B^t, x^t)$ as g^t. It is obvious that the squared ℓ_2 norm terms and the barrier function $\psi(B^t)$ in the objective g^t (5.24), are non-negative. Therefore, we have

$$\lambda Q(W^t) \leq g^t \leq g^0 \tag{C.2}$$

where the second inequality follows from Lemma 25. Now, the function $Q(W^t) = \sum_{i=1}^n (0.5\alpha_i^2 - \log \alpha_i)$, where α_i $(1 \leq i \leq n)$ are the singular values of W^t, is a coercive function of the (non-negative) singular values, and therefore, it has bounded lower level sets [1]. Combining this fact with (C.2), we can immediately conclude that $\exists\ c_0 \in \mathbb{R}$ depending on g^0 and λ, such that $\|W^t\|_F \leq c_0\ \forall\ t$.

Finally, the boundedness of of $\{B^t\}$ follows from the following arguments. First, for Algorithm A1 (see Fig. 5.1), we have that $B^t = H_s\left(W^t X^{t-1}\right)$, where X^{t-1} contains $R_j x^{t-1}$ as its columns. Therefore, by the definition of $H_s(\cdot)$, we have

$$\left\|B^t\right\|_F = \left\|H_s\left(W^t X^{t-1}\right)\right\|_F \leq \left\|W^t X^{t-1}\right\|_F \leq \left\|W^t\right\|_2 \left\|X^{t-1}\right\|_F \tag{C.3}$$

Since, by our previous arguments, W^t and X^{t-1} are both bounded by constants independent of t, we have that the sequence of sparse code matrices $\{B^t\}$, is also bounded. □

We now establish some key optimality properties of the accumulation points of the iterate sequence in Algorithm A1.

Lemma 27. *All the accumulation points of the iterate sequence generated by Algorithm A1 with a given initialization correspond to a common objective value g^*. Thus, they are equivalent in that sense.*

Proof. : Consider the subsequence $\{W^{q_t}, B^{q_t}, x^{q_t}\}$ (indexed by q_t) of the iterate sequence, that converges to the accumulation point (W^*, B^*, x^*). Before proving the lemma, we discuss some simple properties of (W^*, B^*, x^*). First, equation (C.2) implies that $-\log|\det W^{q_t}| \leq (g^0/\lambda)$, for every t. This further implies that $|\det W^{q_t}| \geq e^{-g^0/\lambda} > 0\ \forall\ t$. Therefore, due to the continuity of the function $|\det W|$, the limit W^* of the subsequence is also non-singular, with $|\det W^*| \geq e^{-g^0/\lambda}$. Second, $B^{q_t} = H_s\left(W^{q_t} X^{q_t - 1}\right)$, where X^{q_t-1} is the matrix with $R_j x^{q_t-1}$ $(1 \leq j \leq N)$ as its columns. Thus, B^{q_t} trivially satisfies $\psi(B^{q_t}) = 0$ for every t. Now, $\{B^{q_t}\}$ converges to B^*, which makes B^* the limit of a sequence of matrices, each of which has no more than s non-zeros. Thus, the limit B^* obviously cannot have more than s non-zeros. Therefore,

[1]The lower level sets of a function $f : A \subset \mathbb{R}^n \mapsto \mathbb{R}$ (where A is unbounded) are bounded if $\lim_{t\to\infty} f(z^t) = +\infty$ whenever $\{z^t\} \subset A$ and $\lim_{t\to\infty} \|z^t\| = \infty$.

$\|B^*\|_0 \leq s$, or equivalently $\psi(B^*) = 0$. Finally, since x^{q_t} satisfies the constraint $\|x^{q_t}\|_2^2 \leq C^2$, we have $\chi(x^{q_t}) = 0 \; \forall \; t$. Additionally, since $x^{q_t} \rightarrow x^*$ as $t \rightarrow \infty$, we also have $\|x^*\|_2^2 = \lim_{t\rightarrow\infty} \|x^{q_t}\|_2^2 \leq C^2$. Therefore, $\chi(x^*) = 0$. Now, it is obvious from the above arguments that

$$\lim_{t\rightarrow\infty} \chi(x^{q_t}) = \chi(x^*), \;\; \lim_{t\rightarrow\infty} \psi(B^{q_t}) = \psi(B^*) \tag{C.4}$$

Moreover, due to the continuity of $Q(W)$ at non-singular matrices W, we have $\lim_{t\rightarrow\infty} Q(W^{q_t}) = Q(W^*)$. We now use these limits along with some simple properties of convergent sequences (e.g., the limit of the sum/product of convergent sequences is equal to the sum/product of their limits), to arrive at the following result.

$$\begin{aligned}
&\lim_{t\rightarrow\infty} g(W^{q_t}, B^{q_t}, x^{q_t}) = \\
&\lim_{t\rightarrow\infty} \left\{ \textstyle\sum_{j=1}^{N} \left\| W^{q_t} R_j x^{q_t} - b_j^{q_t} \right\|_2^2 + \nu \left\| A x^{q_t} - y \right\|_2^2 + \lambda \, Q(W^{q_t}) + \psi(B^{q_t}) \right\} \\
&+ \lim_{t\rightarrow\infty} \chi(x^{q_t}) = \sum_{j=1}^{N} \left\| W^* R_j x^* - b_j^* \right\|_2^2 + \nu \left\| A x^* - y \right\|_2^2 + \lambda \, Q(W^*) + \psi(B^*) \\
&= g(W^*, B^*, x^*)
\end{aligned} \tag{C.5}$$

The above result together with the fact (from Lemma 25) that $\lim_{t\rightarrow\infty} g(W^{q_t}, B^{q_t}, x^{q_t}) = g^*$ implies that $g(W^*, B^*, x^*) = g^*$. □

The following lemma establishes partial global optimality with respect to the image, of every accumulation point.

Lemma 28. *Any accumulation point* (W^*, B^*, x^*) *of the iterate sequence generated by Algorithm A1 satisfies*

$$x^* \in \arg\min_{x} \; g\left(W^*, B^*, x\right) \tag{C.6}$$

Proof. : Consider the subsequence $\{W^{q_t}, B^{q_t}, x^{q_t}\}$ (indexed by q_t) of the iterate sequence, that converges to the accumulation point (W^*, B^*, x^*). Then, due to the optimality of x^{q_t} in the image update step of Algorithm A1, we have the following inequality for any t and any $x \in \mathbb{C}^p$.

$$\sum_{j=1}^{N} \left\| W^{q_t} R_j x^{q_t} - b_j^{q_t} \right\|_2^2 + \nu \left\| A x^{q_t} - y \right\|_2^2 + \chi(x^{q_t}) \leq \sum_{j=1}^{N} \left\| W^{q_t} R_j x - b_j^{q_t} \right\|_2^2 + \nu \left\| Ax - y \right\|_2^2 + \chi(x) \qquad \text{(C.7)}$$

Now, by (C.4), we have that $\lim_{t\to\infty} \chi(x^{q_t}) = \chi(x^*)$. We now take the limit $t \to \infty$ on both sides of (C.7) for some fixed $x \in \mathbb{C}^p$, to get the following result

$$\sum_{j=1}^{N} \left\| W^* R_j x^* - b_j^* \right\|_2^2 + \nu \left\| A x^* - y \right\|_2^2 + \chi(x^*) \leq \sum_{j=1}^{N} \left\| W^* R_j x - b_j^* \right\|_2^2 + \nu \left\| Ax - y \right\|_2^2 + \chi(x) \qquad \text{(C.8)}$$

where we have taken the limits term-by-term on both sides of (C.7).

Since the choice of x in (C.8) is arbitrary, equation (C.8) holds for any $x \in \mathbb{C}^p$. Recall that $\psi(B^*) = 0$ and $Q(W^*)$ is finite based on the arguments in the proof of Lemma 27. Therefore, (C.8) implies that $g(W^*, B^*, x^*) \leq g(W^*, B^*, x)\ \forall\ x \in \mathbb{C}^p$. Now, we immediately have the result (C.6) of the Lemma. $\square$

The following lemma will be used to establish that the change between successive image iterates $\|x^t - x^{t-1}\|_2$, converges to 0.

Lemma 29. *Consider the subsequence $\{W^{q_t}, B^{q_t}, x^{q_t}\}$ (indexed by q_t) of the iterate sequence, that converges to the accumulation point (W^*, B^*, x^*). Then, the subsequence $\{x^{q_t-1}\}$ also converges to x^*.*

Proof. : First, by applying Lemma 27, we have that

$$g(W^*, B^*, x^*) = g^* \qquad \text{(C.9)}$$

Next, consider a convergent subsequence $\left\{x^{q_{n_t}-1}\right\}$ of $\{x^{q_t-1}\}$ that converges to say x^{**}. Now, applying the same arguments as in Equation (C.5) (in the proof of Lemma 27), but with respect to the (convergent) subsequence $\left\{x^{q_{n_t}-1}, W^{q_{n_t}}, B^{q_{n_t}}\right\}$, we have that

$$\lim_{t\to\infty} g(W^{q_{n_t}}, B^{q_{n_t}}, x^{q_{n_t}-1}) = g(W^*, B^*, x^{**}) \qquad \text{(C.10)}$$

Now, the monotonic decrease of the objective in Algorithm A1 implies

$$g(W^{q_{n_t}}, B^{q_{n_t}}, x^{q_{n_t}}) \leq g(W^{q_{n_t}}, B^{q_{n_t}}, x^{q_{n_t}-1}) \leq g(W^{q_{n_t}-1}, B^{q_{n_t}-1}, x^{q_{n_t}-1}) \tag{C.11}$$

Taking the limit $t \to \infty$ all through (C.11) and using Lemma 25 (for the extreme left/right limits), and Equation C.10 (for the middle limit), we immediately get that $g(W^*, B^*, x^{**}) = g^*$. This result together with (C.9) implies $g(W^*, B^*, x^{**}) = g(W^*, B^*, x^*)$.

Now, we know by Lemma 28 that

$$x^* \in \underset{x}{\arg\min}\ g\left(W^*, B^*, x\right) \tag{C.12}$$

Furthermore, it was shown in Section 5.3.1.3 of Chapter 5 that if the set of patches in our formulation (P5.1) cover all pixels in the image (always true for the case of periodically positioned overlapping image patches), then the minimization of $g\left(W^*, B^*, x\right)$ with respect to x has a unique solution. Therefore, we have that x^* is the unique minimizer in (C.12). Combining this with the fact that $g(W^*, B^*, x^{**}) = g(W^*, B^*, x^*)$, we get that $x^{**} = x^*$. Since we worked with an arbitrary convergent subsequence $\left\{x^{q_{n_t}-1}\right\}$ (of $\{x^{q_t-1}\}$) in the above proof, we have that x^* is the limit of any convergent subsequence of $\{x^{q_t-1}\}$. Finally, since every convergent subsequence of $\{x^{q_t-1}\}$ converges to x^*, we therefore have that the sequence $\{x^{q_t-1}\}$) itself converges to x^*, which completes the proof. □

Lemma 30. *The iterate sequence* $\{W^t, B^t, x^t\}$ *in Algorithm A1 satisfies*

$$\lim_{t\to\infty} \left\|x^t - x^{t-1}\right\|_2 = 0 \tag{C.13}$$

Proof. : Let the sequence $\{a^t\}$ be such that $a^t \triangleq \|x^t - x^{t-1}\|_2$. We will show below that every convergent subsequence of (the bounded) $\{a^t\}$ converges to 0, thereby implying that the sequence $\{a^t\}$ itself converges to 0 (i.e., there is no other accumulation point for the sequence, except 0).

Now, let us consider a convergent subsequence of $\{a^t\}$ denoted as $\{a^{q_t}\}$. Since the sequence $\{W^{q_t}, B^{q_t}, x^{q_t}\}$ is bounded, there exists a convergent subsequence $\{W^{q_{n_t}}, B^{q_{n_t}}, x^{q_{n_t}}\}$ converging to say (W^*, B^*, x^*). By Lemma 29, we then have that $\left\{x^{q_{n_t}-1}\right\}$ also converges to x^*. Thus, the subsequence $\{a^{q_{n_t}}\}$ with $a^{q_{n_t}} = \|x^{q_{n_t}} - x^{q_{n_t}-1}\|_2$ converges to 0. Since, $\{a^{q_{n_t}}\}$ itself is a subsequence of a convergent sequence, we must have that $\{a^{q_t}\}$ converges to the same limit (0). We have thus shown that zero is the limit of any convergent subsequence of $\{a^t\}$. $\square$

The next property is the partial global optimality of every accumulation point with respect to the sparse code. In order to establish this property, we need the following Lemma.

Lemma 31. *Consider a bounded matrix sequence $\left\{Z^k\right\}$ with $Z^k \in \mathbb{C}^{n\times N}$, that converges to Z^*. Then, every accumulation point of $\left\{H_s(Z^k)\right\}$ belongs to the set $\tilde{H}_s(Z^*)$.*

Proof. : The proof is very similar to that for Lemma 24 in Appendix B.4. $\square$

Lemma 32. *Any accumulation point (W^*, B^*, x^*) of the iterate sequence generated by Algorithm A1 satisfies*

$$B^* \in \underset{B}{\arg\min}\; g\left(W^*, B, x^*\right) \tag{C.14}$$

Moreover, denoting by $X^* \in \mathbb{C}^{n\times N}$ the matrix whose j^th^ *column is $R_j x^*$, for $1 \leq j \leq N$, the above condition can be equivalently stated as*

$$B^* \in \tilde{H}_s(W^* X^*) \tag{C.15}$$

Proof. : Consider the subsequence $\{W^{q_t}, B^{q_t}, x^{q_t}\}$ (indexed by q_t) of the iterate sequence, that converges to the accumulation point (W^*, B^*, x^*). Then, by Lemma 29, $\{x^{q_t-1}\}$ converges to x^*, and the following inequalities hold, where X^{q_t-1} is the matrix with $R_j x^{q_t-1}$ $(1 \leq j \leq N)$ as its columns.

$$B^* = \lim_{t\to\infty} B^{q_t} = \lim_{t\to\infty} H_s\left(W^{q_t} X^{q_t-1}\right) \in \tilde{H}_s(W^* X^*) \tag{C.16}$$

Since $W^{q_t} X^{q_t-1} \to W^* X^*$ as $t \to \infty$, we get the last containment relationship above by applying Lemma 31. While we have proved (C.15), Equation (C.14) now immediately follows by applying the definition of $\tilde{H}_s(\cdot)$ in (C.1). $\square$

The next result pertains to the partial global optimality with respect to W, of every accumulation point in Algorithm A1. We use the following lemma (simple extension of Lemma 1 in [96] to the complex field) to establish the result.

Lemma 33. *Consider a sequence $\{M_k\}$ with $M_k \in \mathbb{C}^{n\times n}$, that converges to M. For each k, let $V_k \Sigma_k R_k^H$ denote a full SVD of M_k. Then, every accumulation point (V, Σ, R) of the sequence $\{V_k, \Sigma_k, R_k\}$ is such that $V\Sigma R^H$ is a full SVD of M. In particular, $\{\Sigma_k\}$ converges to Σ, the $n \times n$ singular value matrix of M.*

Lemma 34. *Any accumulation point (W^*, B^*, x^*) of the iterate sequence generated by Algorithm A1 satisfies*

$$W^* \in \underset{W}{\arg\min}\ g\left(W, B^*, x^*\right) \tag{C.17}$$

Proof. : The proposed Algorithm A1 in Fig. 5.1 involves L alternations between the transform update and sparse coding steps in every outer iteration. At a certain (outer) iteration t, let us denote the intermediate transform update outputs as $\tilde{W}^{(l,t)}$, $1 \leq l \leq L$. Then, $W^t = \tilde{W}^{(L,t)}$. We will only use the sequence $\left\{\tilde{W}^{(1,t)}\right\}$ (i.e., the intermediate outputs for $l = 1$) in the following proof.

Consider the subsequence $\{W^{q_t}, B^{q_t}, x^{q_t}\}$ (indexed by q_t) of the iterate sequence in Algorithm A1, that converges to the accumulation point (W^*, B^*, x^*). Let X^{q_t} be the matrix with $R_j x^{q_t}$ $(1 \leq j \leq N)$ as its columns. Then, $\tilde{W}^{(1,q_t+1)}$ is computed as follows, using the full SVD $V^{q_t}\Sigma^{q_t}\left(R^{q_t}\right)^H$, of $\left(L^{q_t}\right)^{-1} X^{q_t}\left(B^{q_t}\right)^H$, where $\left(L^{q_t}\right)^{-1} = \left(X^{q_t}\left(X^{q_t}\right)^H + 0.5\lambda I\right)^{-1/2}$.

$$\tilde{W}^{(1,q_t+1)} = 0.5R^{q_t}\left(\Sigma^{q_t} + \left(\left(\Sigma^{q_t}\right)^2 + 2\lambda I\right)^{\frac{1}{2}}\right)\left(V^{q_t}\right)^H\left(L^{q_t}\right)^{-1} \tag{C.18}$$

In order to prove the lemma, we will consider the limit $t \to \infty$ in (C.18). In order to take this limit, we need the following results. First, due to the continuity of the matrix square root and matrix inverse functions at positive definite matrices, we have that the following limit holds, where $X^* \in \mathbb{C}^{n\times N}$ has $R_j x^*$ $(1 \leq j \leq N)$ as its columns.

$$\lim_{t\to\infty}\left(L^{q_t}\right)^{-1} = \lim_{t\to\infty}\left(X^{q_t}\left(X^{q_t}\right)^H + 0.5\lambda I\right)^{-1/2} = \left(X^*\left(X^*\right)^H + 0.5\lambda I\right)^{-1/2}$$

Next, defining $L^* \triangleq \left(X^*\left(X^*\right)^H + 0.5\lambda I\right)^{1/2}$, we also have that

$$\lim_{t\to\infty}\left(L^{q_t}\right)^{-1} X^{q_t}\left(B^{q_t}\right)^H = \left(L^*\right)^{-1} X^*\left(B^*\right)^H \tag{C.19}$$

Using the above result and applying Lemma 33, we have that every accumulation point (V^*, Σ^*, R^*) of the sequence $\{V^{q_t}, \Sigma^{q_t}, R^{q_t}\}$ is such that $V^*\Sigma^* (R^*)^H$ is a full SVD of $(L^*)^{-1} X^* (B^*)^H$. Now, consider a convergent subsequence $\{V^{q_{n_t}}, \Sigma^{q_{n_t}}, R^{q_{n_t}}\}$ of $\{V^{q_t}, \Sigma^{q_t}, R^{q_t}\}$, with limit (V^*, Σ^*, R^*). Then, taking the limit $t \to \infty$ in (C.18) along this subsequence, we have

$$W^{**} \triangleq \lim_{t\to\infty} \tilde{W}^{(1,q_{n_t}+1)} = 0.5R^* \left(\Sigma^* + \left((\Sigma^*)^2 + 2\lambda I\right)^{\frac{1}{2}}\right) (V^*)^H (L^*)^{-1}$$

Combining this result with the aforementioned definitions of the square root L^*, and the full SVD $V^*\Sigma^* (R^*)^H$, and applying Proposition 5, we get

$$W^{**} \in \underset{W}{\arg\min}\ g(W, B^*, x^*) \tag{C.20}$$

Finally, applying the same arguments as in the proof of Lemma 27 to the subsequence $\left\{B^{q_{n_t}}, x^{q_{n_t}}, \tilde{W}^{(1,q_{n_t}+1)}\right\}$, we easily get that $g^* = g(W^{**}, B^*, x^*)$. Since, by Lemma 27, we also have that $g^* = g(W^*, B^*, x^*)$, we get $g(W^*, B^*, x^*) = g(W^{**}, B^*, x^*)$, which together with (C.20) immediately establishes the required result (C.17). □

The following lemma establishes that every accumulation point of the iterate sequence in Algorithm A1 is a critical point of the objective $g(W, B, x)$. All derivatives or sub-differentials are computed with respect to the real and imaginary parts of the corresponding variables/vectors/matrices below.

Lemma 35. *Every accumulation point (W^*, B^*, x^*) of the iterate sequence generated by Algorithm A1 is a critical point of the objective $g(W, B, x)$ satisfiying*

$$0 \in \partial g(W^*, B^*, x^*) \tag{C.21}$$

Proof. : Consider the subsequence $\{W^{q_t}, B^{q_t}, x^{q_t}\}$ (indexed by q_t) of the iterate sequence, that converges to the accumulation point (W^*, B^*, x^*). By Lemmas 34, 28, and 32, we have that

$$W^* \in \underset{W}{\arg\min}\ g(W, B^*, x^*) \tag{C.22}$$

$$x^* \in \underset{x}{\arg\min}\ g(W^*, B^*, x) \tag{C.23}$$

$$B^* \in \underset{B}{\arg\min}\ g(W^*, B, x^*) \tag{C.24}$$

The function $g(W, B, x)$ is continuously differentiable with respect to W at non-singular points W. Since W^* is a (non-singular) partial global minimizer in (C.22), we have as a necessary condition that $\nabla_W g\left(W^*, B^*, x^*\right) = 0$. Next, recall the statement in Section 5.4.1, that a necessary condition for $z \in \mathbb{R}^q$ to be a minimizer of some function $\phi : \mathbb{R}^q \mapsto (-\infty, +\infty]$ is that z satisfies $0 \in \partial\phi(z)$. Now, since x^* and B^* are partial global minimizers in (C.23) and (C.24), respectively, we therefore have that $0 \in \partial g_x\left(W^*, B^*, x^*\right)$ and $0 \in \partial g_B\left(W^*, B^*, x^*\right)$, respectively. It is also easy to derive these conditions directly using the definition (Definition 2) of the sub-differential.

Finally, (see Proposition 3 in [201]) the subdifferential ∂g at (W^*, B^*, x^*) satisfies

$$\partial g\left(W^*, B^*, x^*\right) = \nabla_W g\left(W^*, B^*, x^*\right) \times \partial g_B\left(W^*, B^*, x^*\right) \times \partial g_x\left(W^*, B^*, x^*\right) \tag{C.25}$$

Now, using the preceding results, we easily have that $0 \in \partial g\left(W^*, B^*, x^*\right)$ above. Thus, every accumulation point in Algorithm A1 is a critical point of the objective. □

The following two lemmas establish pairwise partial local optimality of the accumulation points in Algorithm A1. Here, $X^* \in \mathbb{C}^{n \times N}$ is the matrix with $R_j x^*$ as its columns.

Lemma 36. *Every accumulation point (W^*, B^*, x^*) of the iterate sequence generated by Algorithm A1 is a partial minimizer of the objective $g(W, B, x)$ with respect to (W, B), in the sense of (5.30), for sufficiently small $dW \in \mathbb{C}^{n \times n}$, and all $\Delta B \in \mathbb{C}^{n \times N}$ in the union of the regions R1 and R2 in Theorem 3. Furthermore, if $\|W^* X^*\|_0 \leq s$, then the ΔB in (5.30) can be arbitrary.*

Proof. : Consider the subsequence $\{W^{q_t}, B^{q_t}, x^{q_t}\}$ (indexed by q_t) of the iterate sequence, that converges to the accumulation point (W^*, B^*, x^*). First, by Lemmas 32 and 34, we have

$$B^* \in \tilde{H}_s(W^* X^*) \tag{C.26}$$

$$2W^* X^* \left(X^*\right)^H - 2B^* \left(X^*\right)^H + \lambda W^* - \lambda \left(W^*\right)^{-H} = 0 \tag{C.27}$$

where the second equality follows from the first order conditions for partial global optimality of W^* in (C.17). The accumulation point (W^*, B^*, x^*) also satisfies $\psi(B^*) = 0$ and $\chi(x^*) = 0$.

Now, considering perturbations $dW \in \mathbb{C}^{n\times n}$, and $\Delta B \in \mathbb{C}^{n\times N}$, we have that

$$g(W^* + dW, B^* + \Delta B, x^*) = \|W^*X^* - B^* + (dW)X^* - \Delta B\|_F^2 \\ + \nu \|Ax^* - y\|_2^2 + 0.5\lambda \|W^* + dW\|_F^2 - \lambda \log |\det (W^* + dW)| + \psi(B^* + \Delta B) \tag{C.28}$$

In order to prove the condition (5.30) in Theorem 3, it suffices to consider *sparsity preserving* perturbations ΔB, that is $\Delta B \in \mathbb{C}^{n\times N}$ such that $B^*+\Delta B$ has sparsity $\leq s$. Otherwise $g(W^*+dW, B^*+\Delta B, x^*) = +\infty > g(W^*, B^*, x^*)$ trivially. Therefore, we only consider sparsity preserving ΔB in the following, for which $\psi(B^* + \Delta B) = 0$ in (C.28).

Since the image x^* is fixed here, we can utilize (C.26) and (C.27), and apply similar arguments as in the proof of Lemma 9 (equations (40)-(43)) in [96], to simplify the right hand side in (C.28). The only difference is that the matrix transpose $(\cdot)^T$ operations in [96] are replaced with Hermitian $(\cdot)^H$ operations here, and the operation $\langle Q, R\rangle$ involving two matrices (or, vectors) Q and R in [96] is redefined [2] here as

$$\langle Q, R\rangle \triangleq Re\left\{\mathrm{tr}(QR^H)\right\} \tag{C.29}$$

Upon such simplifications, we can conclude [96] that $\exists \epsilon' > 0$ depending on W^* such that whenever $\|dW\|_F < \epsilon'$, we have

$$g(W^* + dW, B^* + \Delta B, x) \geq g(W^*, B^*, x) - 2\langle W^*X^* - B^*, \Delta B\rangle \tag{C.30}$$

The term $-\langle W^*X^* - B^*, \Delta B\rangle$ above is trivially non-negative for ΔB in region R1 in Theorem 3, and therefore, $g(W^*+dW, B^*+\Delta B, x) \geq g(W^*, B^*, x)$ for $\Delta B \in$ R1. It is also easy to see that any sparsity preserving ΔB in region R2 in Theorem 3 will have its support (i.e., non-zero locations) contained in the support of $B^* \in \tilde{H}_s(W^*X^*)$. Therefore, $\langle W^*X^* - B^*, \Delta B\rangle = 0$ for any sparsity preserving $\Delta B \in$ R2. This result together with (C.30) implies $g(W^* + dW, B^* + \Delta B, x) \geq g(W^*, B^*, x)$ for any $\Delta B \in$ R2. Finally, if $\|W^*X^*\|_0 \leq s$, then by (C.26), $W^*X^* - B^* = 0$ in (C.30). Therefore, in this

[2]We include the $Re(\cdot)$ operation in the definition here, which allows for simpler notations in the rest of the proof. However, the $\langle Q, R\rangle$ defined in (C.29) is no longer the conventional inner product.

case, the ΔB in (5.30) can be arbitrary. $\square$

Lemma 37. *Every accumulation point (W^*, B^*, x^*) of the iterate sequence generated by Algorithm A1 is a partial minimizer of the objective $g(W, B, x)$ with respect to (B, x), in the sense of* (5.31)*, for all $\tilde{\Delta} x \in \mathbb{C}^p$, and all $\Delta B \in \mathbb{C}^{n\times N}$ in the union of the regions R1 and R2 in Theorem 3. Furthermore, if $\|W^* X^*\|_0 \leq s$, then the ΔB in* (5.31) *can be arbitrary.*

Proof. : Consider the subsequence $\{W^{q_t}, B^{q_t}, x^{q_t}\}$ (indexed by q_t) of the iterate sequence, that converges to the accumulation point (W^*, B^*, x^*). It follows from Lemmas 32 and 28 that $\psi(B^*) = \chi(x^*) = 0$. Now, considering perturbations $\Delta B \in \mathbb{C}^{n\times N}$ whose columns are denoted as Δb_j $(1 \leq j \leq N)$, and $\tilde{\Delta} x \in \mathbb{C}^p$, we have that

$$
\begin{aligned}
g(W^*, B^* + \Delta B, x^* + \tilde{\Delta} x) = \sum_{j=1}^{N} \left\| W^* R_j x^* - b_j^* + W^* R_j \tilde{\Delta} x - \Delta b_j \right\|_2^2 \\
+ \lambda\, Q(W^*) + \nu \left\| Ax^* - y + A\tilde{\Delta} x \right\|_2^2 + \psi(B^* + \Delta B) + \chi(x^* + \tilde{\Delta} x)
\end{aligned}
\tag{C.31}
$$

In order to prove the condition (5.31) in Theorem 3, it suffices to consider *sparsity preserving* perturbations ΔB, that is $\Delta B \in \mathbb{C}^{n\times N}$ such that $B^* + \Delta B$ has sparsity $\leq s$. It also suffices to consider *energy preserving* perturbations $\tilde{\Delta} x$, which are such that $\left\| x^* + \tilde{\Delta} x \right\|_2 \leq C$. For any other ΔB or $\tilde{\Delta} x$, $g(W^*, B^* + \Delta B, x^* + \tilde{\Delta} x) = +\infty > g(W^*, B^*, x^*)$ trivially. Therefore, we only consider the energy/sparsity preserving perturbations in the following, for which $\psi(B^* + \Delta B) = 0$ and $\chi(x^* + \tilde{\Delta} x) = 0$ in (C.31). Now, upon expanding the squared ℓ_2 terms in (C.31), and dropping non-negative perturbation terms, we get

$$
\begin{aligned}
g(W^*, B^* + \Delta B, x^* + \tilde{\Delta} x) \geq g(W^*, B^*, x^*) - 2 \sum_{j=1}^{N} \left\langle W^* R_j x^* - b_j^*, \Delta b_j \right\rangle \\
+ 2 \sum_{j=1}^{N} \left\langle W^* R_j x^* - b_j^*, W^* R_j \tilde{\Delta} x \right\rangle + 2\nu \left\langle Ax^* - y, A\tilde{\Delta} x \right\rangle
\end{aligned}
\tag{C.32}
$$

where the $\langle \cdot, \cdot \rangle$ notation is as defined in (C.29). Now, using identical arguments as were made in the case of (C.30), it is clear that the term

$-2\sum_{j=1}^{N}\left\langle W^{*}R_{j}x^{*}-b_{j}^{*},\Delta b_{j}\right\rangle \geq 0$ in (C.32) for all sparsity preserving $\Delta B \in$ R1 $\cup$ R2. Furthermore, since by Lemma 28, x^{*} is a global minimizer of $g(W^{*},B^{*},x)$, it must satisfy the Normal equation (5.16) for some (unique) $\mu \geq 0$. Using these arguments, (C.32) simplifies to

$$g(W^{*},B^{*}+\Delta B,x^{*}+\tilde{\Delta}x) \geq g(W^{*},B^{*},x^{*})-2\mu\left\langle x^{*},\tilde{\Delta}x\right\rangle \tag{C.33}$$

Now, if $\left\|x^{*}\right\|_{2}<C$, then (the optimal) $\mu=0$ above, and therefore, $g(W^{*},B^{*}+\Delta B,x^{*}+\tilde{\Delta}x) \geq g(W^{*},B^{*},x^{*})$ for arbitrary $\tilde{\Delta}x \in \mathbb{C}^{p}$, and $\Delta B \in$ R1 $\cup$ R2. The alternative scenario is the case $\left\|x^{*}\right\|_{2}=C$, i.e., when $\mu>0$. Since we are considering energy preserving perturbations $\tilde{\Delta}x$, we have $\left\|x^{*}+\tilde{\Delta}x\right\|_{2}^{2} \leq C^{2}=\left\|x^{*}\right\|_{2}^{2}$, which implies $-2\left\langle x^{*},\tilde{\Delta}x\right\rangle \geq \left\|\tilde{\Delta}x\right\|_{2}^{2} \geq 0$. Combining this result with (C.33), we again have (now for the $\mu>0$ case) that $g(W^{*},B^{*}+\Delta B,x^{*}+\tilde{\Delta}x) \geq g(W^{*},B^{*},x^{*})$ for arbitrary $\tilde{\Delta}x \in \mathbb{C}^{p}$, and $\Delta B \in$ R1 $\cup$ R2. $\square$

C.2 Proofs of Theorems 4 and 5

The proofs of Theorems 4 and 5 are very similar to that for Theorem 3. We only discuss some of the minor differences, as follows.

First, in the case of Theorem 4, the main difference in the proof is that the non-negative barrier function $\psi(B)$ and the operator $H_{s}(\cdot)$ (in the proof of Theorem 3) are replaced by the non-negative penalty $\eta^{2}\sum_{j=1}^{N}\left\|b_{j}\right\|_{0}$ and the operator $\hat{H}_{\eta}^{1}(\cdot)$, respectively. Moreover, the mapping $\tilde{H}_{s}(\cdot)$ defined in (C.1) for the proof of Theorem 3, is replaced by the matrix-to-set mapping $\hat{H}_{\eta}(\cdot)$ (for the proof of Theorem 4) defined as

$$\left(\hat{H}_{\eta}(Z)\right)_{ij}=\begin{cases} 0 & , \ \left|Z_{ij}\right|<\eta \\ \{Z_{ij},\,0\} & , \ \left|Z_{ij}\right|=\eta \\ Z_{ij} & , \ \left|Z_{ij}\right|>\eta \end{cases} \tag{C.34}$$

By thus replacing the relevant functions and operators, the various steps in the proof of Theorem 3 can be easily extended to the case of Theorem 4. As such, Theorem 4 mainly differs from Theorem 3 in terms of the definition of the set of allowed (local) perturbations ΔB. In particular, the proofs of partial local optimality of accumulation points in Theorem 4 are easily extended from the aforementioned proofs for Lemmas 36 and 37, by using the techniques and inequalities mentioned in Appendix E of [96].

Finally, the main difference between the proofs of Theorems 3 and 5 is that the non-negative $\lambda\, Q(W)$ penalty in the former is replaced by the barrier function $\varphi(W)$ (that enforces the unitary property, and keeps W^t always bounded) in the latter. Otherwise, the proof techniques are very similar for the two cases.

APPENDIX D

APPROXIMATION ERROR IN ONLINE ALGORITHM

Here, we calculate the error introduced by the approximation (7.6) in the transform update step of our online learning algorithm. Let us denote $L_t^{-1}\Theta_t$, $L_t^{-1}y_t x_t^T$, and $\left(L_t^{-1} - L_{t-1}^{-1}\right)\Theta_{t-1}$ as M_t, z_t, and Υ_t, respectively. Then, by (7.5), we have

$$M_t = (1 - t^{-1})M_{t-1} + \Upsilon_t + z_t \tag{D.1}$$

Equation (7.6) introduces an approximation to M_t, and then computes the SVD of the approximate matrix. Let us denote the approximate matrix as $\hat{M}_t$. The SVD of $\hat{M}_t$ is computed via a rank-1 update to the SVD of $(1 - t^{-1})\hat{M}_{t-1}$. Thus, we have by (7.6) that

$$\hat{M}_t = (1 - t^{-1})\hat{M}_{t-1} + z_t \tag{D.2}$$

Subtracting (D.2) from (D.1) and denoting $M_t - \hat{M}_t$ as E_t, yields

$$E_t = (1 - t^{-1})(E_{t-1} + \Upsilon_t) \tag{D.3}$$

Assuming $E_1 = 0$, equation (D.3) implies that $E_t = \sum_{j=2}^{t} \frac{j-1}{t}\Upsilon_j$. ■

APPENDIX E

RESULTS FOR PROOFS IN CHAPTER 8

The notations and assumptions in this Appendix are as defined in Chapter 8.

E.1 Lipschitz Continuity of $u^{(1)}(y, W)$

Lemma 38. *Let Assumption A1 hold. Then, the function* $u^{(1)}(y, W) = \|Wy - H_s(Wy)\|_2^2$ *is uniformly Lipschitz with respect to* W *on the bounded set* $S \triangleq \{W \in \mathbb{R}^{n\times n} : \|W\|_2 \leq 1\}$.

Proof: Consider a bounded open set A containing S, such that $W \in A$ satisfies $\|W\|_2 \leq c$ for some finite $c > 1$. Let us choose two matrices W_1 and W_2 from A. To simplify notation in this proof, we denote $u^{(1)}(y, W_i) = a_i^2$, $a_i \geq 0$, $i = 1, 2$. Then, clearly

$$\left|u^{(1)}(y, W_1) - u^{(1)}(y, W_2)\right| = |a_1 + a_2| \cdot |a_1 - a_2| \tag{E.1}$$

Since $|a_1 + a_2| \leq 4c$, we need to only (Lipschitz) bound $|a_1 - a_2|$. Let us define $\Delta \triangleq W_1 - W_2$. Then, we have

$$a_1 = \left\|W_2 y + \Delta y - H_s(W_2 y + \Delta y)\right\|_2 \tag{E.2}$$

Let Γ be a diagonal matrix, with entries either 0 or 1. The 1's are at locations corresponding to the support (indices of the non-zero locations) of $H_s(W_2 y + \Delta y)$. Then, applying the triangle inequality to (E.2), we have

$$a_1 \leq \left\|W_2 y - \Gamma W_2 y\right\|_2 + \left\|\Delta y - \Gamma \Delta y\right\|_2$$

$$a_1 \geq \left\|W_2 y - \Gamma W_2 y\right\|_2 - \left\|\Delta y - \Gamma \Delta y\right\|_2$$

Now, by Lemma 39 in Appendix E.2, it is clear that there exists an $\epsilon > 0$ depending on $W_2 y$ (or, depending on W_2, for fixed y) such that whenever $\left\|\Delta\right\|_2 < \epsilon$ (i.e., $\left\|\Delta y\right\|_2 < \epsilon$), then the support of $H_s(W_2 y + \Delta y)$ either coincides with, or contains the support of one of the optimal sparse codes in $\tilde{H}_s(W_2 y)$.

It follows that for $\left\|\Delta\right\|_2 < \epsilon$, we have that Γ is also the support of a code in $\tilde{H}_s(W_2 y)$ [1], and because all the codes in $\tilde{H}_s(W_2 y)$ provide the same sparsification error, we have $\left\|W_2 y - \Gamma W_2 y\right\|_2 = \left\|W_2 y - H_s(W_2 y)\right\|_2$. Furthermore, $\left\|\Delta y - \Gamma \Delta y\right\|_2 \leq \left\|\Delta y\right\|_2 \leq \left\|\Delta\right\|_2$. Combining the inequalities for a_1 with the above results, we obtain

$$-\left\|\Delta\right\|_2 \leq a_1 - a_2 \leq \left\|\Delta\right\|_2 = \left\|W_1 - W_2\right\|_2 \tag{E.3}$$

Combining (E.3) and (E.1), we obtain

$$\left|u^{(1)}(y, W_1) - u^{(1)}(y, W_2)\right| \leq 4c \left\|W_1 - W_2\right\|_2 \tag{E.4}$$

whenever $\left\|W_1 - W_2\right\|_2 < \epsilon$. Since W_2 was arbitrarily chosen, and ϵ is a function of W_2 only, (E.4) implies that $u^{(1)}(y, W)$ is locally Lipschitz on the open set A. Finally, since S is a compact subset of A ($u^{(1)}(y, W)$ is also bounded on S), it follows from standard results [202] that $u^{(1)}(y, W)$ is Lipschitz on S. $\blacksquare$

E.2 Support of a Thresholded Perturbed Vector

For a vector h, we let $\beta_j(h)$ denote the magnitude of the j^{th} largest element (magnitude-wise) of h.

Lemma 39. *Let us consider a vector $\alpha^* \in \mathbb{R}^n$, and a perturbation $\delta \in \mathbb{R}^n$ of this vector. Let s be a given sparsity level. Then, there exists an $\epsilon > 0$ such that the support of $H_s(\alpha^* + \delta)$ contains the support of one of the optimal codes in $\tilde{H}_s(\alpha^*)$, whenever $\|\delta\|_2 < \epsilon$. Furthermore, except in the (degenerate) case when $\beta_s(\alpha^*) = 0$, the support of $H_s(\alpha^* + \delta)$ coincides with the support of one of the optimal codes in $\tilde{H}_s(\alpha^*)$, whenever $\|\delta\|_2 < \epsilon$.*

[1] In the trivial case that the s^{th} largest magnitude element of $W_2 y$ is zero, Γ contains the entire support of the singleton $\tilde{H}_s(W_2 y) = W_2 y$ (Appendix E.2).

Proof: First, let us assume that $\tilde{H}_s(\alpha^*)$ (the set of optimal projections of α^* onto the s-ℓ_0 ball) is a singleton, and $\beta_s(\alpha^*) > 0$, so that $\beta_s(\alpha^*) - \beta_{s+1}(\alpha^*) > 0$. Then, whenever $\|\delta\|_\infty < (\beta_s(\alpha^*) - \beta_{s+1}(\alpha^*))/2$ holds, we have that $H_s(\alpha^* + \delta)$ has the same support set (non-zero locations) as $H_s(\alpha^*) = \tilde{H}_s(\alpha^*)$.

Next, when $\tilde{H}_s(\alpha^*)$ is a singleton, but $\beta_s(\alpha^*) = 0$ (and $\alpha^* \neq 0$), let γ be the magnitude of the non-zero element of α^* of smallest magnitude. Then, for any $\delta \in \mathbb{R}^n$ satisfying $\|\delta\|_\infty < \gamma/2$, we have that the support of $H_s(\alpha^*) = \tilde{H}_s(\alpha^*)$ is contained in the support of $H_s(\alpha^* + \delta)$.

Finally, let us assume $\tilde{H}_s(\alpha^*)$ is not a singleton (there are ties). Let us define $a \in \mathbb{R}^{n-1}$ as follows

$$a_j \triangleq \frac{\beta_j(\alpha^*) - \beta_{j+1}(\alpha^*)}{2} \quad 1 \leq j \leq n-1 \tag{E.5}$$

Let us set ϵ to be the smallest non-zero element of a. Then, it is easy to show that the support of $H_s(\alpha^* + \delta)$ coincides with the support of one of the optimal codes in $\tilde{H}_s(\alpha^*)$ whenever $\|\delta\|_\infty < \epsilon$.

If the a computed using α^* in (E.5) is a zero vector, then all elements of α^* must have identical magnitude. In this case, the lemma trivially holds for any $\epsilon > 0$. ■

E.3 Useful Theorems

Here, we list some theorems relevant to the convergence analysis of our online algorithm. The first theorem is from the perturbation theory of singular value decompositions (cf. [203], [204], and Chapter 15 of [205] and references therein).

Theorem 10. *Let matrices $A \in \mathbb{R}^{n \times n}$ and $\tilde{A} \in \mathbb{R}^{n \times n}$, with $\tilde{A} = A + E$, where E is a perturbation matrix. Let $A = U \Sigma V^T$ and $\tilde{A} = \tilde{U} \tilde{\Sigma} \tilde{V}^T$ be the respective SVDs with the respective singular values σ_i and $\tilde{\sigma}_i$ arranged in decreasing order for $1 \leq i \leq n$. Then, we have*

$$|\tilde{\sigma}_i - \sigma_i| \leq \|E\|_2, \ \forall \, i \tag{E.6}$$

Furthermore, for any index i, define

$$\eta_i = \min\left\{ \left(\min_{j:j\neq i} |\tilde{\sigma}_i - \sigma_j| \right), \tilde{\sigma}_i + \sigma_i \right\} \tag{E.7}$$

If $\eta_i > 0$, then

$$\min_{\alpha\in\{-1,+1\}} \sqrt{\|u_i\alpha - \tilde{u}_i\|_2^2 + \|v_i\alpha - \tilde{v}_i\|_2^2} \leq \sqrt{2}\frac{\sqrt{\|r\|_2^2 + \|s\|_2^2}}{\eta_i} \tag{E.8}$$

where u_i, v_i, $\tilde{u}_i$, and $\tilde{v}_i$ denote the i^{th} *columns of U, V, $\tilde{U}$, and $\tilde{V}$, respectively, and $r = A\tilde{v}_i - \tilde{\sigma}_i\tilde{u}_i$, $s = A^T\tilde{u}_i - \tilde{\sigma}_i\tilde{v}_i$. Specifically, $\|r\|_2 \leq \|E\|_2$ and $\|s\|_2 \leq \|E\|_2$.*

When the perturbation E is small, Theorem 10 indicates that corresponding singular values of A and $\tilde{A} = A + E$ are close. In order for the corresponding singular vectors of A and $\tilde{A}$ to be close to each other as well (up to ± 1 scaling), the constant η_i in (E.7) needs to be positive for each i. This is true, for example, when A has full rank, and has distinct singular values, and E is sufficiently small.

The next two Theorems (cf. [163, 161] and references therein) are used in the proof of proposition 11. In theorem 11, the expectation $\mathbb{E}_X[\cdot]$ is calculated with respect to the probability measure on χ, and X_1, X_2,.. are random vectors distributed according to this probability measure.

Theorem 11. *Let $\mathcal{G} = \{g_\theta : \chi \mapsto \mathbb{R}, \theta \in \Xi\}$ be a set of measurable functions indexed by a bounded subset Ξ of $\mathbb{R}^d$. Suppose that there exists a constant M such that $|g_{\theta_1}(x) - g_{\theta_2}(x)| \leq M\|\theta_1 - \theta_2\|_2$ for every $\theta_1, \theta_2 \in \Xi$ and $x \in \chi$. Then, $\mathcal{G}$ is P-Donsker (see [163]). For any g in $\mathcal{G}$, define $\mathbb{P}_t g \triangleq \frac{1}{t}\sum_{j=1}^t g(X_j)$, $\mathbb{P}g \triangleq \mathbb{E}_X[g(X)]$, and $\mathbb{H}_t g \triangleq \sqrt{t}(\mathbb{P}_t g - \mathbb{P}g)$. Let us also suppose that for all g, $\mathbb{P}g^2 < \delta^2$ and $\|g\|_\infty < C$ and that the random elements X_1, X_2,... are Borel-measurable. Then, we have*

$$\mathbb{E}\left[\sup_{g\in\mathcal{G}} |\mathbb{H}_t g|\right] = O(1) \tag{E.9}$$

The following theorem [161, 206] is on the convergence of a stochastic sequence.

Theorem 12. *Let $(\Omega, \mathcal{F}, P)$ be a measurable probability space, r_t, for $t \geq 0$, be a realization of a stochastic process and $\mathcal{F}_t$ be the filtration determined by the past information at time t. Let us define*

$$\delta_t = \begin{cases} 1 & \text{if } \mathbb{E}[r_{t+1} - r_t | \mathcal{F}_t] > 0 \\ 0 & \text{otherwise} \end{cases}$$

If for all t, $r_t \geq 0$, and $\sum_{t=1}^{\infty} \mathbb{E}[\delta_t \cdot (r_{t+1} - r_t)] < \infty$, then r_t is a quasi-martingale and converges almost surely. Moreover,

$$\sum_{t=1}^{\infty} \Big| \mathbb{E}[r_{t+1} - r_t | \mathcal{F}_t] \Big| < \infty \quad a.s. \tag{E.10}$$

The next result is on the limit of a real sequence [20].

Theorem 13. *Let $\{b_k\}$ and $\{d_k\}$ be two sequences of real numbers such that $\forall$ k, we have $b_k \geq 0$, $d_k \geq 0$, $\sum_{k=1}^{\infty} b_k d_k < \infty$, and $\sum_{k=1}^{\infty} b_k = \infty$. Furthermore, $\exists$ $C > 0$ such that $|d_{k+1} - d_k| < C b_k$ $\forall$ k. Then, $\lim_{k \to \infty} d_k = 0$.*

Finally, the following result is on the directional differentiability of optimal value functions [207, 208].

Theorem 14. *Let $h : \mathbb{R}^p \times \mathbb{R}^q \mapsto \mathbb{R}$. Suppose that for all $x \in \mathbb{R}^p$, the function $h(x, \cdot)$ is differentiable, and that h and $\nabla_u h(x, u)$ the derivative of $h(x, \cdot)$ are continuous on $\mathbb{R}^p \times \mathbb{R}^q$. Let $l(u)$ be the optimal value function defined as $l(u) = \min_{x \in A} h(x, u)$, where A is a compact subset of $\mathbb{R}^p$. Then, we have that $l(u)$ is directionally differentiable. Furthermore, if for $u_0 \in \mathbb{R}^q$, $h(\cdot, u_0)$ has a unique minimizer x_0, then $l(u)$ is differentiable at u_0 and $\nabla_u l(u_0) = \nabla_u h(x_0, u_0)$.*

E.4 On the Existence of $\nabla g^{(1)}(W)$

In the following Lemma 41, we discuss the existence and continuity of $\nabla g^{(1)}(\cdot)$. The Assumptions A1-A4 stated in Chapter 8 hold. First, the following lemma establishes conditions under which the operation $\tilde{H}_s(\cdot)$ produces a singleton set.

Lemma 40. *Let $y \in \mathbb{R}^n$ be distributed over the n-dimensional unit sphere $\{y \in \mathbb{R}^n : \|y\|_2 = 1\}$, with an absolutely continuous probability measure. Consider a matrix $W \in \mathbb{R}^{n \times n}$, that has full rank. Then $\tilde{H}_s(Wy)$ is a singleton with probability 1.*

Proof: In order for $\tilde{H}_s(Wy)$ to not be a singleton (i.e., non-unique transform sparse coding solution), a necessary condition is that at least two entries of the vector Wy have the same magnitude, i.e., $w_i y = w_j y$ or $w_i y = -w_j y$, where w_i and w_j are distinct rows of W. This event corresponds to $(w_i \pm w_j)y = 0$. Since W is full rank, $(w_i \pm w_j) \neq 0$. Therefore, the non-singleton event occurs when y lies on one of the two $n-1$ dimensional hyperplanes characterized by $(w_i \pm w_j)z = 0$. However, the intersection of each these $n-1$ dimensional hyperplanes with the n-dimensional unit sphere is a set of zero Lebesgue measure, and hence, because of the absolute continuity of the distribution of y, $\mathbb{P}\{(w_i \pm w_j)y = 0\} = 0$. This easily implies that the probability that any two entries (any $i \neq j$) of the vector Wy have the same magnitude is zero, or that $\tilde{H}_s(Wy)$ is a singleton with probability 1. ■

Lemma 41. *Consider a matrix $W \in \mathbb{R}^{n \times n}$, that has full rank, and is bounded ($\|W\|_2 \leq 1$). Then, $\nabla g^{(1)}(\cdot)$ exists, and is continuous at W.*

Proof: For $y \in \mathbb{R}^n$ with $\|y\|_2 = 1$, we have

$$u^{(1)}(y, W) = \min_{x : \|x\|_0 \leq s} \|Wy - x\|_2^2 \tag{E.11}$$

The set of optimal $\hat{x}$ above is characterized by $\tilde{H}_s(Wy)$. Similar to the bound in (8.8), we can conclude that any optimal $\hat{x}$ satisfies $\|\hat{x}\|_2 \leq 1$. Therefore, without loss of generality, we can consider the minimization in (E.11) over the set $A \triangleq \{x \in \mathbb{R}^n : \|x\|_0 \leq s, \|x\|_2 \leq 1\}$. It is easy to verify that A is compact.

Now, we apply Theorem 14 in Appendix E.3 to prove that $u^{(1)}(y, \cdot)$ is continuously differentiable at (full rank, bounded) W with probability 1. For any y, for which $\tilde{H}_s(Wy)$ is a singleton, it is easy to check that all conditions in Theorem 14 are satisfied with the function h in Theorem 14 defined as $\|Wy - x\|_2^2$, and the compact set A defined as above. Therefore, $\nabla_W u^{(1)}(y, W)$ exists (at the full rank, bounded W) and is equal [2]

[2]It is also easy to derive $\nabla_W u^{(1)}(y, W)$ (since $\tilde{H}_s(Wy)$ is a singleton) starting from the

to $2Wyy^T - 2H_s(Wy)y^T$. It can be easily verified (using similar arguments as in Appendix E.2, and the proof of Lemma 38) that the gradient $2Wyy^T - 2H_s(Wy)y^T$ (which is bounded in Frobenius norm by 2) is continuous at W, when $\tilde{H}_s(Wy)$ is a singleton. For our full rank and bounded W, since $\tilde{H}_s(Wy)$ is a singleton with probability 1 (by Lemma 40), we therefore have that the function $u^{(1)}(y, \cdot)$ is continuously differentiable at W with probability 1.

Based on the above results, the following equation holds.

$$\nabla g^{(1)}(W) = \nabla \mathbb{E}_y[u^{(1)}(y, W)] = \mathbb{E}_y \left[\nabla_W u^{(1)}(y, W) \right] \tag{E.12}$$

Thus, $\nabla g^{(1)}(W)$ exists for any $W \in \mathbb{R}^{n \times n}$ that is full rank with $\|W\|_2 \leq 1$. It is easy to show that it is continuous (since the derivative of $u^{(1)}(y, \cdot)$ is continuous at W with probability 1) at each such W. ■

definition of the derivative as a limit, even without using Theorem 14.

APPENDIX F

PROOFS FOR CHAPTER 9

F.1 Convergence Analysis of DOSLIST

The proof of the convergence rate in Theorem 6 is similar to that for FISTA [11], except that we utilize the block Lipschitz properties, as defined by equation (9.7). The various following Lemmas will lead to the result in Theorem 6. We provide detailed proofs in some places, and indicate the similarities to the FISTA proof otherwise.

We let $\hat{g}(A, X) = \mu \left\|A\right\|_1 + \xi \left\|X\right\|_1$ denote the non-smooth part of the objective in (P9.1). We denote the right hand side of the inequality (9.7) as $\hat{p}(A, A', X, X')$. Thus, equation (9.7) can be written as $f(A', X') \leq \hat{p}(A, A', X, X')$. Then, it is clear that $F(A', X') \leq \hat{p}(A, A', X, X') + \hat{g}(A', X')$, and we denote the right hand side of this inequality as $p(A, A', X, X')$. We further let

$$\{u(A, X),\, v(A, X)\} = \arg\min_{A', X'} p(A, A', X, X') \tag{F.1}$$

Note that the optimization above is only performed with respect to the off-diagonal entries of A'. The diagonal entries of both A and A' are fixed to be zero. The solution to the above problem has the following form.

$$u(A, X) = S_{\mu/L_A}\left(A - \frac{1}{L_A}\nabla_A f(A, X)\right) \tag{F.2}$$

$$v(A, X) = S_{\xi/L_X}\left(X - \frac{1}{L_X}\nabla_X f(A, X)\right) \tag{F.3}$$

The diagonal entries of $u(A, X)$ are automatically zero above. Note that the update of A^k and X^k in Figure 9.1 is based on the formulae in equations (F.2) and (F.3), respectively. We use the preceding notations in the following lemma, which we prove.

Lemma 42. *Let $A \in \mathbb{R}^{n\times n}$ with zero diagonal, $X \in \mathbb{R}^{n\times N}$, and $L_A > 0$, $L_X > 0$ be such that*

$$F(u(A,X), v(A,X)) \leq p(A, u(A,X), X, v(A,X)) \tag{F.4}$$

Then, for any $A' \in \mathbb{R}^{n\times n}$ with zero diagonal, and $X' \in \mathbb{R}^{n\times N}$, we have that

$$\begin{aligned} F(A',X') - F(u(A,X), v(A,X)) &\geq \frac{L_A}{2}\left\|u(A,X) - A\right\|_F^2 \\ &+ L_A \langle A - A', u(A,X) - A\rangle + L_X \langle X - X', v(A,X) - X\rangle \\ &+ \frac{L_X}{2}\left\|v(A,X) - X\right\|_F^2 \end{aligned} \tag{F.5}$$

Proof: Based on convexity of f and $\hat{g}$, we have the following inequalities.

$$f(A',X') \geq f(A,X) + \langle A' - A, \nabla_A f(A,X)\rangle + \langle X' - X, \nabla_X f(A,X)\rangle \tag{F.6}$$

$$\hat{g}(A',X') \geq \hat{g}(u(A,X), v(A,X)) + \langle A' - u(A,X), \delta_A\rangle + \langle X' - v(A,X), \delta_X\rangle \tag{F.7}$$

where δ_A and δ_X are the A and X components of some subgradient of $\hat{g}(A,X)$ at the point $(u(A,X), v(A,X))$ satisfying the following inequalities.

$$\nabla_A f(A,X) + L_A (u(A,X) - A) + \delta_A = 0 \tag{F.8}$$

$$\nabla_X f(A,X) + L_X (v(A,X) - X) + \delta_X = 0 \tag{F.9}$$

The existence of such a δ_A and δ_X follows from the optimality condition for (F.1). We get a lower bound on $F(A',X')$ by adding equations (F.6) and (F.7).

$$\begin{aligned} F(A',X') &\geq \langle A' - A, \nabla_A f(A,X)\rangle + \langle X' - X, \nabla_X f(A,X)\rangle \qquad \text{(F.10)} \\ &+ f(A,X) + \hat{g}(u(A,X), v(A,X)) + \langle A' - u(A,X), \delta_A\rangle + \langle X' - v(A,X), \delta_X\rangle \end{aligned}$$

Moreover, equation (F.4) implies that

$$F(A',X') - F(u(A,X), v(A,X)) \geq F(A',X') - p(A, u(A,X), X, v(A,X)) \tag{F.11}$$

Using the definition of the $p(\cdot)$ function, we get that

$p(A, u(A,X), X, v(A,X))$ satisfies the following equation.

$$\begin{aligned} p(A, u(A,X), X, v(A,X)) &= \langle \nabla_A f(A,X), u(A,X) - A\rangle \\ &+ \frac{L_A}{2}\|u(A,X) - A\|_F^2 + \frac{L_X}{2}\|v(A,X) - X\|_F^2 + f(A,X) \\ &+ \langle \nabla_X f(A,X), v(A,X) - X\rangle + \hat{g}(u(A,X), v(A,X)) \end{aligned} \tag{F.12}$$

Thus, plugging equations (F.10) and (F.12) into (F.11), we get

$$\begin{aligned} F(A', X') - F(u(A,X), v(A,X)) &\geq -\frac{L_X}{2}\|v(A,X) - X\|_F^2 \\ &+ \langle X' - v(A,X), \nabla_X f(A,X) + \delta_X\rangle - \frac{L_A}{2}\|u(A,X) - A\|_F^2 \\ &+ \langle A' - u(A,X), \nabla_A f(A,X) + \delta_A\rangle \end{aligned} \tag{F.13}$$

Now, using the property of δ_A and δ_X from equations (F.8) and (F.9), (F.13) becomes

$$\begin{aligned} F(A', X') - F(u(A,X), v(A,X)) &\geq -\frac{L_X}{2}\|v(A,X) - X\|_F^2 \\ &- L_X\langle X' - v(A,X), v(A,X) - X\rangle - \frac{L_A}{2}\|u(A,X) - A\|_F^2 \\ &- L_A\langle A' - u(A,X), u(A,X) - A\rangle \end{aligned} \tag{F.14}$$

The right hand side of the above equation is algebraically identical to the right hand side of the required bound (F.5). ■

Note that equation (F.4) is always satisfied when the constants L_A and L_X obey (9.7).

Lemma 43. *Let $\{A^k\}$, $\{X^k\}$, $\{Q^k\}$, $\{R^k\}$ denote the iterate sequences generated by the DOSLIST algorithm for data Z. Further, let (A^*, X^*) denote any minimizer of Problem (P9.1). Then, $\forall k \geq 1$, we have*

$$t_k^2 c_k - t_{k+1}^2 c_{k+1} \geq \frac{L_X}{2}\left\|d_X^{k+1}\right\|_F^2 - \frac{L_X}{2}\left\|d_X^k\right\|_F^2 + \frac{L_A}{2}\left\|d_A^{k+1}\right\|_F^2 - \frac{L_A}{2}\left\|d_A^k\right\|_F^2 \tag{F.15}$$

where $c_k = F(A^k, X^k) - F(A^*, X^*)$, and $d_X^k = t_k X^k - (t_k - 1)X^{k-1} - X^*$, $d_A^k = t_k A^k - (t_k - 1)A^{k-1} - A^*$.

Proof: We apply Lemma 42 at $(A', X') = (A^k, X^k)$, $(A, X) =$

(Q^{k+1}, R^{k+1}). We also apply it at $(A', X') = (A^*, X^*)$, $(A, X) = (Q^{k+1}, R^{k+1})$. We then get the following two inequalities.

$$\begin{aligned} c_k - c_{k+1} \geq & \frac{L_A}{2} \left\| A^{k+1} - Q^{k+1} \right\|_F^2 + \frac{L_X}{2} \left\| X^{k+1} - R^{k+1} \right\|_F^2 \\ & + L_A \left\langle Q^{k+1} - A^k, A^{k+1} - Q^{k+1} \right\rangle \\ & + L_X \left\langle R^{k+1} - X^k, X^{k+1} - R^{k+1} \right\rangle \qquad \text{(F.16)} \\ -c_{k+1} \geq & \frac{L_A}{2} \left\| A^{k+1} - Q^{k+1} \right\|_F^2 + \frac{L_X}{2} \left\| X^{k+1} - R^{k+1} \right\|_F^2 \\ & + L_A \left\langle Q^{k+1} - A^*, A^{k+1} - Q^{k+1} \right\rangle \\ & + L_X \left\langle R^{k+1} - X^*, X^{k+1} - R^{k+1} \right\rangle \qquad \text{(F.17)} \end{aligned}$$

We multiply (F.16) by $t_{k+1}^2 - t_{k+1}$ and (F.17) by t_{k+1}, and add the two resulting inequalities to get

$$\begin{aligned} t_k^2 c_k - t_{k+1}^2 c_{k+1} \geq & \frac{L_A}{2} \left\| t_{k+1}(A^{k+1} - Q^{k+1}) \right\|_F^2 \\ & + L_A t_{k+1} \left\langle t_{k+1} Q^{k+1} - (t_{k+1} - 1) A^k - A^*, A^{k+1} - Q^{k+1} \right\rangle \\ & + \frac{L_X}{2} \left\| t_{k+1}(X^{k+1} - R^{k+1}) \right\|_F^2 \\ & + L_X t_{k+1} \left\langle t_{k+1} R^{k+1} - (t_{k+1} - 1) X^k - X^*, X^{k+1} - R^{k+1} \right\rangle \\ & \qquad \text{(F.18)} \end{aligned}$$

where we have used the fact that $t_{k+1}^2 - t_{k+1} = t_k^2$. For each of the L_A and L_X based components on the right hand side (RHS) of (F.18), we use the Pythagorean relation $\|b - a\|_F^2 + 2\langle b - a, a - c\rangle = \|b - c\|_F^2 - \|a - c\|_F^2$ [1]. The first two terms on the RHS of (F.18) are L_A-based, and $b = \sqrt{(L_A/2)}\, t_{k+1} A^{k+1}$, $a = \sqrt{(L_A/2)}\, t_{k+1} Q^{k+1}$, $c = \sqrt{(L_A/2)}\, ((t_{k+1} - 1) A^k + A^*)$. The last two terms on the RHS of (F.18) are L_X-based, and the corresponding $b = \sqrt{(L_X/2)}\, t_{k+1} X^{k+1}$, $a = \sqrt{(L_X/2)}\, t_{k+1} R^{k+1}$, and $c = \sqrt{(L_X/2)}\, ((t_{k+1} - 1) X^k + X^*)$.

After applying the Pythagorean relations to the RHS of (F.18), we simplify the RHS of the resulting inequality by substituting for Q^{k+1} and R^{k+1} from equations (9.13) and (9.14), respectively. This gives the required result. ■

The scalar sequence $\{t_k\}$ in Figure 9.1 is known to satisfy $t_k \geq (k+1)/2$ $\forall\, k \geq 1$ [11]. The following lemma is also directly from [11].

[1]The terms on the RHS of (F.18) correspond to the left hand side of this Pythagorean relation.

Lemma 44. *Let $\{a_k, b_k\}$ be positive sequences of real numbers with*

$$a_k - a_{k+1} \geq b_{k+1} - b_k \quad \forall\, k \geq 1 \tag{F.19}$$

with $a_1 + b_1 \leq \gamma$ for some $\gamma > 0$. Then, $a_k \leq \gamma \;\forall\, k \geq 1$.

F.1.1 Proof of Theorem 6

Similar to [11], we will use the preceding lemmas to briefly prove Theorem 6. We use the notations defined in Lemma 43. Let $a_k = t_k^2 c_k$, and $b_k = \frac{L_X}{2}\left\|d_X^k\right\|_F^2 + \frac{L_A}{2}\left\|d_A^k\right\|_F^2$. Then, by Lemma 43, we have that $a_k - a_{k+1} \geq b_{k+1} - b_k \;\forall k \geq 1$. Now, to apply Lemma 44, we need to find a $\gamma > 0$ such that $a_1 + b_1 \leq \gamma$, where $a_1 = c_1$ and $b_1 = \frac{L_X}{2}\left\|X^1 - X^*\right\|_F^2 + \frac{L_A}{2}\left\|A^1 - A^*\right\|_F^2$. Using Lemma 42 with $(A', X') = (A^*, X^*)$, and $(A, X) = (A^0, X^0)$, we get

$$\begin{aligned} F(A^*, X^*) - F(A^1, X^1) \geq & \frac{L_A}{2}\left\|A^1 - A^*\right\|_F^2 - \frac{L_A}{2}\left\|A^0 - A^*\right\|_F^2 \\ & + \frac{L_X}{2}\left\|X^1 - X^*\right\|_F^2 - \frac{L_X}{2}\left\|X^0 - X^*\right\|_F^2 \end{aligned} \tag{F.20}$$

where we used the fact that $u(A^0, X^0) = A^1$ and $v(A^0, X^0) = X^1$. The right hand side above is identical to the right hand side of equation (F.5) (with simple rearrangement of terms/completion of squares). Equation (F.20) directly implies that the required

$$\gamma = \frac{L_A}{2}\left\|A^0 - A^*\right\|_F^2 + \frac{L_X}{2}\left\|X^0 - X^*\right\|_F^2 \tag{F.21}$$

Thus, using Lemma 44 along with the lower bound on t_k from [11], we get the required bound $c_k \leq 4\gamma/(k+1)^2$. ■

F.2 Proof of Lemma 5

We have that the modified function is $\widetilde{f}(A, X, \alpha Z) = \left\|(I + A)\alpha Z - X\right\|_F^2 + \frac{\alpha^2 \eta}{4}\left\|A + A^T\right\|_F^2$. If we substitute $X = \alpha X''$, then it immediately follows that

$$\widetilde{f}(A, X, \alpha Z) = \widetilde{f}(A, \alpha X'', \alpha Z) = \alpha^2 f(A, X'', Z) \tag{F.22}$$

Next, we look at equation (9.7) for the modified function $\widetilde{f}(A, X, \alpha Z)$. If we replace (substitute for) X and X' with αX and $\alpha X'$ respectively, we have

$$\widetilde{f}(A', \alpha X', \alpha Z) \leq \widetilde{f}(A, \alpha X, \alpha Z) + \frac{L_A^{'}}{2} \left\| A' - A \right\|_F^2 + \frac{L_X^{'} \alpha^2}{2} \left\| X' - X \right\|_F^2 + \left\langle \nabla_A \widetilde{f}(A, \alpha X, \alpha Z), A' - A \right\rangle + \alpha \left\langle \nabla_X \widetilde{f}(A, \alpha X, \alpha Z), X' - X \right\rangle \quad \text{(F.23)}$$

Here, $L_A^{'}$ and $L_X^{'}$ are the modified Lipschitz constants. Similar to (9.7), equation (F.23) holds for arbitrary matrices A, A', X, X'. Now, using the definitions of the gradients from equations (9.9) and (9.8), we get $\nabla_A \widetilde{f}(A, \alpha X, \alpha Z) = \alpha^2 \nabla_A f(A, X, Z)$ and $\nabla_X \widetilde{f}(A, \alpha X, \alpha Z) = \alpha \nabla_X f(A, X, Z)$. Using this result and equation (F.22), it is clear that every term in equation (F.23) except $L_A^{'} \left\| A' - A \right\|_F^2 / 2$ is a scaled version (scaled by α^2) of the corresponding term in (9.7). Thus, upon dividing the entire equation (F.23) by α^2, we get

$$f(A', X', Z) \leq f(A, X, Z) + \frac{L_A^{'}}{2\alpha^2} \left\| A' - A \right\|_F^2 + \frac{L_X^{'}}{2} \left\| X' - X \right\|_F^2 + \langle \nabla_A f(A, X, Z), A' - A \rangle + \langle \nabla_X f(A, X, Z), X' - X \rangle \quad \text{(F.24)}$$

Finally, comparing equation (F.24) to (9.7), we conclude the relation $L_X^{'} = L_X$ and $L_A^{'} = \alpha^2 L_A$.

F.3 Convergence Analysis of DOSLAM

We will first show the convergence of the objective function in DOSLAM. Then, we will prove that every accumulation point of the iterates is a fixed point of the algorithm. Next, we will establish that the iterates in DOSLAM converge to an equivalence class of accumulation points. Finally, we show that every accumulation point of the iterates (i.e., fixed point of algorithm) is a local minimizer of the objective $g(A, X)$. These results together establish Theorem 7.

For our analysis, we define $\tilde{H}_s(Z + AZ)$ to be the set of all optimal solutions (projections) in Problem (9.15).

F.3.1 Convergence of Objective

Lemma 46 provides the convergence of the objective for DOSLAM. We will use the following Lemma 45 to prove Lemma 46.

Lemma 45. *Let $\{Q^i\}$ be the iterate sequence generated by the ISTA-based transform update step (equation* (9.17)*) of DOSLAM, with fixed sparse code X. Then, the objective sequence $\{g(Q^i, X)\}$ is monotone decreasing. If two consecutive elements in this objective sequence are identical, then ISTA has reached a fixed point, which is also a global minimizer of the transform update step.*

Proof: The monotonicity of the objective sequence is known for ISTA [11], but we prove it in our setting for completeness. Fix an $i \geq 1$, and consider equation (9.7) with $X' = X$ (sparse X), $A = Q^{i-1}$, and L_A replaced by L. We then have

$$\begin{aligned} g(A', X) \leq & \left\langle \nabla_A f(Q^{i-1}, X), A' - Q^{i-1} \right\rangle + \frac{L}{2} \left\| A' - Q^{i-1} \right\|_F^2 \\ & + g(Q^{i-1}, X) + \mu \left\| A' \right\|_1 - \mu \left\| Q^{i-1} \right\|_1 \end{aligned} \quad \text{(F.25)}$$

Let us denote the right hand side of the above equation as $P_L\left(A', Q^{i-1}, X\right)$. It is easy to see that $P_L\left(A', Q^{i-1}, X\right)$ is strictly/strongly convex with respect to A', and thus has a unique minimizer with respect to A'. The ISTA solution in equation (9.17) satisfies

$$Q^i = \arg\min_{A'} P_L\left(A', Q^{i-1}, X\right) \quad \text{(F.26)}$$

Thus, we have

$$P_L\left(Q^i, Q^{i-1}, X\right) \leq P_L\left(Q^{i-1}, Q^{i-1}, X\right) = g(Q^{i-1}, X) \quad \text{(F.27)}$$

Due to strict convexity of $P_L\left(A', Q^{i-1}, X\right)$, the first inequality above is an equality if and only if $Q^i = Q^{i-1}$. Combining (F.27) with (F.25) (with $A' = Q^i$), we get $g(Q^i, X) \leq g(Q^{i-1}, X)$. Thus, the objective sequence is monotone decreasing. Moreover, $g(Q^i, X) = g(Q^{i-1}, X)$ if and only if $Q^i = Q^{i-1}$. It is clear from equation (9.17) that if $Q^i = Q^{i-1}$, then Q^i becomes a fixed point for the ISTA. Since ISTA converges to a global minimizer (see e.g., [11]) of its problem, this implies that a fixed point Q^i is also a global minimizer of the transform update step. ■

Lemma 46. *Let $\{A^k, X^k\}$ denote the iterate sequence generated by the DOSLAM algorithm for (P9.2) with data Z. Then, the sequence of objective function values $\{g(A^k, X^k)\}$ is monotone decreasing, and converges to a finite value, say $g^* = g^*(A^0, X^0)$.*

Proof: In the transform update step of DOSLAM, ISTA [165, 11] (or, MFISTA [170]) is employed. ISTA ensures a monotone decrease of the objective value. Thus, $g(A^{k+1}, X^k) \leq g(A^k, X^k)$. In the sparse coding step of DOSLAM, we obtain an exact solution for X with fixed A. Thus, again the objective can only decrease, i.e., $g(A^{k+1}, X^{k+1}) \leq g(A^{k+1}, X^k)$. Combining the results for the two steps, we have $g(A^{k+1}, X^{k+1}) \leq g(A^k, X^k)$. Since, the objective sequence is monotone decreasing and lower bounded by 0, it converges. ■

Lemma 46 shows that the objective sequence $\{g(A^k, X^k)\}$ is monotone decreasing. In fact, it is easy to show using Lemma 45 that the objective sequence is strictly monotone decreasing (until convergence). This is because for every k, $g(A^{k+1}, X^k) \leq g(A^k, X^k)$ (see proof of Lemma 45), with equality if and only if $A^{k+1} = A^k$. Moreover, if $A^{k+1} = A^k$, then $X^{k+1} = X^k$ [2], and $g(A^{k+1}, X^{k+1}) = g(A^{k+1}, X^k) = g(A^k, X^k)$. Thus, if any two consecutive values in the objective sequence are identical, it implies that the algorithm has reached a fixed point (note that once $(A^{k+1}, X^{k+1}) = (A^k, X^k)$, the algorithm remains at that point), and that the objective has converged. In general, the objective need not converge (i.e., iterates need not reach a fixed point) in a finite number of iterations. We are more interested in analyzing this general case.

F.3.2 Existence of Accumulation Point

The next lemma shows the boundedness of the iterates and existence of an accumulation point for the iterates.

[2] This rule is trivially satisfied (for $k \geq 1$) due to the way the DOSLAM Algorithm is written, except perhaps for the case when the superscript $k = 0$ (in the rule). In the latter case, if $A^1 = A^0$, and assuming X^0 is an optimal solution in (9.15) with $A = A^0$, we can (optimally) enforce $X^1 = X^0$ (as an additional rule in this case) in Fig. 9.2. This means that the algorithm has already reached a fixed point (the initial (A^0, X^0) is a fixed point), and therefore, no more iterations are performed. All convergence results hold true for this degenerate case.

Lemma 47. *The iterate sequence $\left\{A^k, X^k\right\}$ generated by the DOSLAM algorithm is bounded, and it has at least one accumulation point.*

Proof: The existence of a convergent subsequence for a bounded sequence is a standard result. Therefore, a bounded sequence has at least one accumulation point. We now prove the boundedness of the iterates. Let us denote $g(A^k, X^k)$ as g^k for simplicity. We then have the boundedness of $\left\{A^k\right\}$ as follows (assume $\mu \neq 0$).

$$\mu \left\|A^k\right\|_1 \leq g^k \leq g^0 \tag{F.28}$$

where the first inequality above follows from the non-negativity of the various terms in the objective g^k, and the second inequality above follows from Lemma 46. We also easily have the following inequalities.

$$\left\|X^k\right\|_F - \left\|(I + A^k)Z\right\|_F \leq \left\|(I + A^k)Z - X^k\right\|_F \leq g^k \leq g^0$$

The above result implies

$$\left\|X^k\right\|_F \leq g^0 + \left\|Z + A^k Z\right\|_F \leq g^0 + \|Z\|_F + \sigma_1 \left\|A^k\right\|_F \tag{F.29}$$

where σ_1 is the largest singular value of the transformed data Z. The boundedness of $\left\{X^k\right\}$ now follows from the fact that $\left\|A^k\right\|_F \leq \left\|A^k\right\|_1 \leq g^0/\mu$. ■

F.3.3 Accumulation Points are Fixed Points

Next, we show in Lemma 49 that any accumulation point of the DOSLAM iterates is a fixed point. We need the following Lemma 48 for that result.

Lemma 48. *Let h be the function mapping defined by the M iterations of ISTA in Fig. 9.2, i.e., at any iteration $k \geq 1$ of DOSLAM, $A^k = h(A^{k-1}, X^{k-1})$, where X^{k-1} is the sparse code matrix, and A^{k-1} is the ISTA initialization at iteration k. Then, the mapping h is continuous with respect to (A^{k-1}, X^{k-1}).*

Proof: Looking at equation (9.17) in the DOSLAM algorithm of Figure 9.2, we can see that for a specific inner ISTA iteration i, the gradient

$\nabla_A f(Q^{i-1}, X^{k-1})$ is linear (by equation (9.9)) with respect to (Q^{i-1}, X^{k-1}). Hence, the term $Q^{i-1} - \frac{1}{L}\nabla_A f(Q^{i-1}, X^{k-1})$ is linear and therefore, continuous, with respect to (Q^{i-1}, X^{k-1}). The soft-thresholding function $S_{\mu/L}(R)$ is also continuous with respect to its argument R. Since the composition of the continuous soft-thresholding and gradient-based steps remains continuous, the Q^i in equation (9.17) varies continuously with respect to (Q^{i-1}, X^{k-1}). Using such an argument successively (one ISTA iteration at a time), we can conclude that $Q^M = A^k$ also varies continuously with respect to $(Q^0, X^{k-1}) = (A^{k-1}, X^{k-1})$. Thus, the mapping h is continuous. ■

Lemma 49. *Any accumulation point of the iterate sequence $\{A^k, X^k\}$ generated by DOSLAM is a fixed point.*

Proof: By Lemma 47, the DOSLAM iterate sequence has at least one accumulation point. Let $\{A^{q_k}, X^{q_k}\}$ be a subsequence of the DOSLAM iterate sequence converging to some accumulation point (A^*, X^*). We then have that

$$X^* = \lim_{k\to\infty} X^{q_k} = \lim_{k\to\infty} \hat{H}_s(Z + A^{q_k}Z) \in \tilde{H}_s(Z + A^*Z) \tag{F.30}$$

which follows from our previous Lemma 24 in Appendix B.4. We now define a function $g'(A, X) = g(A, X) - \psi(X)$, where $\psi(X)$ is the barrier function defined in Section 9.4.3. Then, $g'(A, X)$ is continuous in its arguments. Moreover, for the subsequence $\{A^{q_k}, X^{q_k}\}$ and its accumulation point, the barrier function $\psi(X) = 0$. Since, by Lemma 46, the objective converges for DOSLAM, we have that $g^* = \lim_{k\to\infty} g(A^{q_k}, X^{q_k}) = \lim_{k\to\infty} g(A^{q_k+1}, X^{q_k})$ $= \lim_{k\to\infty} g(A^{q_k+1}, X^{q_k+1})$. Moreover,

$$\begin{aligned}\lim_{k\to\infty} g(A^{q_k}, X^{q_k}) &= \lim_{k\to\infty} g'(A^{q_k}, X^{q_k}) + \lim_{k\to\infty} \psi(X^{q_k}) \\ &= g'(A^*, X^*) + 0 = g(A^*, X^*)\end{aligned} \tag{F.31}$$

where we have used the continuity of g'. Thus, $g^* = g(A^*, X^*)$. We also have that

$$\begin{aligned}\lim_{k\to\infty} g(A^{q_k+1}, X^{q_k}) &= \lim_{k\to\infty} g'(A^{q_k+1}, X^{q_k}) + \lim_{k\to\infty} \psi(X^{q_k}) \\ &= \lim_{k\to\infty} g'(h(A^{q_k}, X^{q_k}), X^{q_k}) = g'(h(A^*, X^*), X^*) \\ &= g(h(A^*, X^*), X^*)\end{aligned} \tag{F.32}$$

where we have used the continuity of both g' and h. Combining the above result with preceding ones, we get

$$g(h(A^*, X^*), X^*) = g^* = g(A^*, X^*) \tag{F.33}$$

This implies that the objective has not decreased after the M iterations of ISTA (equation (9.17)), with fixed sparse code X^*, and initialization A^*. Since, the objective remains constant over the M iterations of ISTA, we have by Lemma 45 that A^* is a fixed point of ISTA satisfying $h(A^*, X^*) = A^*$. Thus, when the accumulation point (A^*, X^*) is fed as initial input to the DOSLAM algorithm, the transform update step produces A^*. Subsequently, performing sparse coding with $h(A^*, X^*) = A^*$ again gives rise to X^* (see footnote 2 in Appendix F.3.1). Thus, repeating the transform update and sparse coding steps on (A^*, X^*) leads to no modification. Therefore, this accumulation point is a fixed point of DOSLAM. ■

F.3.4 Other Properties of the Iterate Sequence

We now present some other interesting properties of the iterate sequence for DOSLAM.

Lemma 50. *If a subsequence $\{A^{q_k}, X^{q_k}\}$ of the DOSLAM iterate sequence converges to (A^*, X^*), then the subsequence $\{A^{q_k+1}\}$ also converges to A^*.*

Proof: Given an iterate (A^{q_k}, X^{q_k}), we can write $A^{q_k+1} = h(A^{q_k}, X^{q_k})$, where h is the continuous mapping defined in Lemma 48 for ISTA. We can now use the continuity of h to obtain

$$\lim_{k\to\infty} A^{q_k+1} = \lim_{k\to\infty} h(A^{q_k}, X^{q_k}) = h(A^*, X^*) \tag{F.34}$$

where we have used the property that $\{A^{q_k}, X^{q_k}\}$ converges to (A^*, X^*). Finally, from (F.33), we have that $h(A^*, X^*) = A^*$. ■

In Lemma 50, the accumulation points of the sequence $\{X^{q_k+1}\}$ all lie in the (finite) set $\tilde{H}_s(Z + A^*Z)$. This follows easily from Lemma 24 of Appendix B.4.

Lemma 51. *The DOSLAM iterate sequences $\left\{A^k, X^k\right\}$ satisfy the property* *that as $k \to \infty$,*

$$\left\|A^{k+1} - A^k\right\|_F \to 0 \tag{F.35}$$

Proof: Define a sequence $\left\{b^k\right\}$ as $b^k = \left\|A^{k+1} - A^k\right\|_F$. We will show below that every convergent subsequence of $\left\{b^k\right\}$ converges to 0, thereby implying that the sequence $\left\{b^k\right\}$ itself converges to 0 (the required result).

Now, let us consider a convergent subsequence of $\left\{b^k\right\}$ denoted as $\{b^{q_k}\}$. Since the sequence $\{A^{q_k}, X^{q_k}\}$ is bounded, there exists a convergent subsequence of this sequence, which we denote as $\{A^{q_{n_k}}, X^{q_{n_k}}\}$. Let $\{A^{q_{n_k}}, X^{q_{n_k}}\}$ converge to (A^*, X^*). By Lemma 50, we then have that $\{A^{q_{n_k}+1}\}$ also converges to A^*. Thus, the subsequence $\{b^{q_{n_k}}\}$ with $b^{q_{n_k}} = \|A^{q_{n_k}+1} - A^{q_{n_k}}\|_F$ converges to 0. Since, $\{b^{q_{n_k}}\}$ itself is a subsequence of a convergent sequence, we must have that $\{b^{q_k}\}$ converges to the same limit (0). We have thus shown that zero is the limit of any particular convergent subsequence of $\left\{b^k\right\}$. ■

Similar to Lemma 51, we can also have the following result for the sparse code sequence in DOSLAM.

$$\lim_{k\to\infty} \left\|X^{k+1} - X^k\right\|_F = 0 \tag{F.36}$$

The above result holds under the extra assumption that the accumulation points of $\left\{A^k\right\}$ give rise to unique sparse codes (in (9.15)). The proof of (F.36) is similar to that for Lemma 51.

Lemma 51 also implies that for every (i, j), $\left|A_{ij}^{k+1} - A_{ij}^k\right| \to 0$ as $k \to \infty$. Although the fact that the difference between successive iterates converges to zero is an interesting property, it by itself does not guarantee that the iterate sequence converges to a unique accumulation point.

F.3.5 Equivalence of Accumulation Points

The following lemma guarantees that the accumulation points are all equivalent in terms of their objective value.

Lemma 52. *Given a particular initial (A^0, X^0), the accumulation points of the DOSLAM iterate sequence $\left\{A^k, X^k\right\}$ all correspond to the same objective value. Thus, they are equivalent in that sense.*

Proof: Based on equation (F.33), we have that any accumulation point

(A^*, X^*) of the DOSLAM iterate sequence satisfies $g(A^*, X^*) = g^*$, with g^* being the limit of $\{g(A^k, X^k)\}$, as defined in Lemma 46. ■

Lemma 52 implies that for a particular initial (A^0, X^0), the iterate sequence in DOSLAM converges to an equivalence class of accumulation points.

F.3.6 Local Optimality of Accumulation Points

The following lemma establishes that any accumulation point (i.e., fixed point by Lemma 49) of the iterates is a local minimizer of $g(A, X)$. Since, the accumulation points are all equivalent in terms of their cost, Lemma 53 implies that they are equally good local minimizers.

Lemma 53. *Every fixed point (A, X) of the DOSLAM algorithm is a (local) minimizer of the objective $g(A, X)$ of Problem (P9.2). Specifically, (A, X) satisfies*

$$g(A + dA, X + \Delta X) \geq g(A, X) \tag{F.37}$$

for all $dA \in \mathbb{R}^{n\times n}$ with a zero diagonal, and all $\Delta X \in \mathbb{R}^{n\times N}$ satisfying at least one of the following conditions: 1) tr $\left\{(Z + AZ - X)\Delta X^T\right\} \leq 0$; 2) $\|\Delta X\|_\infty < \min_i \{\beta_s(U_i) : \|U_i\|_0 > s\}$, where $U = (I + A)Z$. Moreover, if $\|U_i\|_0 \leq s\,\forall\, i$, then ΔX can be arbitrary.

Proof: Since (A, X) is a fixed point of DOSLAM, we have $h(A, X) = A$, where h is the continuous mapping defined in Lemma 48 for ISTA. Moreover, $g(h(A, X), X) = g(A, X)$, implying that the objective does not change after the M iterations of ISTA, with fixed X and initialization A. Since by Lemma 45. the objective within ISTA is monotone decreasing, $g(h(A, X), X) = g(A, X)$ implies that the objective is in fact constant over the M ISTA iterations involved here. Therefore, by Lemma 45, A is a global minimizer of the convex transform update step with fixed X. From standard convex optimization theory, we know A satisfies $0 \in \partial_A\, g(A, X)$, i.e., there exists a subgradient e of the function $\|A\|_1$, at point A, such that

$$G \odot (2(I + A)ZZ^T - 2XZ^T + \eta A + \eta A^T + \mu\, e) = 0 \tag{F.38}$$

where the matrix G was defined in equation (9.9). Additionally, from the fixed point property, we also easily have

$$X \in \tilde{H}_s(Z + AZ) \tag{F.39}$$

Now, given such a (A, X), consider perturbations $dA \in \mathbb{R}^{n\times n}$ with zero main diagonal, and $\Delta X \in \mathbb{R}^{n\times N}$. We are interested in the relationship between $g(A+dA, X+\Delta X)$ and $g(A, X)$. It suffices to consider ΔX such that $X+\Delta X$ has s-sparse columns. Otherwise the barrier function $\psi(X + \Delta X) = +\infty$, and $g(A + dA, X + \Delta X) > g(A, X)$ trivially. Therefore, in the rest of this proof, we let $\Delta X \in \mathbb{R}^{n\times N}$ be such that $X + \Delta X$ is (at most) s-sparse per column. Then, we have

$$\begin{aligned} g(A + dA, X + \Delta X) = \|(I + A)Z - X + (dA)Z - \Delta X\|_F^2 + \mu \|A + dA\|_1 \\ + \frac{\eta}{4} \left\|A + A^T + dA + dA^T\right\|_F^2 \end{aligned} \tag{F.40}$$

Expanding the two Frobenius norm terms above using the trace inner product $\langle Q, R\rangle \triangleq \mathrm{tr}(QR^T)$, and dropping the non-negative terms $\|(dA)Z - \Delta X\|_F^2$ and $\left\|dA + dA^T\right\|_F^2$, we obtain

$$\begin{aligned} g(A + dA, X + \Delta X) \geq \|(I + A)Z - X\|_F^2 + \frac{\eta}{4} \left\|A + A^T\right\|_F^2 + \mu \|A + dA\|_1 \\ + 2 \langle Z + AZ - X, (dA)Z - \Delta X\rangle + \frac{\eta}{2} \left\langle A + A^T, dA + dA^T\right\rangle \end{aligned} \tag{F.41}$$

Since dA has a zero main diagonal here, we can use (F.38) to simplify (F.41) as

$$\begin{aligned} g(A + dA, X + \Delta X) \geq \|Z + AZ - X\|_F^2 + \frac{\eta}{4} \left\|A + A^T\right\|_F^2 + \mu \|A + dA\|_1 \\ - \langle \mu\, e, dA\rangle - \langle (I + A)Z - X, \Delta X\rangle \end{aligned} \tag{F.42}$$

Next, from the standard definition of the subgradient, we have $\mu \|A + dA\|_1 - \langle \mu\, e, dA\rangle \geq \mu \|A\|_1$. Substituting this result in (F.42), we get

$$g(A + dA, X + \Delta X) \geq g(A, X) - \langle (I + A)Z - X, \Delta X\rangle \tag{F.43}$$

Thus, we have the optimality condition $g(A+dA, X+\Delta X) \geq g(A, X)$ for any $dA \in \mathbb{R}^{n\times n}$ with a zero main diagonal, and for any $\Delta X \in \mathbb{R}^{n\times N}$ satisfying $\langle (I + A)Z - X, \Delta X\rangle \leq 0$. The rest of the lemma follows from (F.39), and by applying the same techniques as in the proof of Lemma 23 in Appendix B.3. Specifically, for $U = (I + A)Z$, if ΔX (ΔX such that $X + \Delta X$ has s-sparse columns) is such that $\|\Delta X\|_\infty < \min_i \{\beta_s(U_i) : \|U_i\|_0 > s\}$, and $dA \in \mathbb{R}^{n\times n}$ with a zero main diagonal, then $\langle (I + A)Z - X, \Delta X\rangle = 0$ in (F.43), and the condition (F.37) follows for such perturbations. ■

F.4 Estimates for Lipschitz constants

As mentioned in a previous footnote (footnote 2 in Chapter 9), condition (9.7) follows from the condition that $\langle \nabla_A f(A', X') - \nabla_A f(A, X),\, A' - A\rangle + \langle \nabla_X f(A', X') - \nabla_X f(A, X),\, X' - X\rangle \leq L_A \|A' - A\|_F^2 + L_X \|X' - X\|_F^2$. We will now derive estimates for L_A and L_X. We know from the standard Cauchy Schwartz inequality that

$$\begin{aligned}\langle \nabla_X f(A', X') - \nabla_X f(A, X),\, X' - X\rangle \leq \\ \|\nabla_X f(A', X') - \nabla_X f(A, X)\|_F \|X' - X\|_F\end{aligned} \tag{F.44}$$

By equation (9.8), we can bound $\|\nabla_X f(A', X') - \nabla_X f(A, X)\|_F$ as

$$\begin{aligned}\|\nabla_X f(A', X') - \nabla_X f(A, X)\| &= 2\|X' - X - A'Z + AZ\| \\ &\leq 2\|X' - X\|_F + 2\sigma_1 \|A' - A\|_F\end{aligned} \tag{F.45}$$

where σ_1 is the largest singular value of Z. Similar to (F.44), we also have the inequality $\langle \nabla_A f(A', X') - \nabla_A f(A, X),\, A' - A\rangle \leq \|\nabla_A f(A', X') - \nabla_A f(A, X)\|_F \|A' - A\|_F$. In this case, using equation (9.9), we get

$$\begin{aligned}\nabla_A f(A', X') - \nabla_A f(A, X) = G \odot \left(2\left(A' - A\right) ZZ^T\right) \\ - G \odot \left(2\left(X' - X\right) Z^T\right) + \eta G \odot \Big(\left(A' - A\right) + \left(A' - A\right)^T\Big)\end{aligned} \tag{F.46}$$

Note that the parameter η scales with the scaling of Z as discussed in Section 9.4. Thus, we can write $\eta = \eta_0\, \sigma_1^2$. Now, using (F.46) and the triangle inequality, it is easy to show that $\|\nabla_A f(A', X') - \nabla_A f(A, X)\|_F \leq (2\sigma_1^2 + 2\eta)\|A' - A\|_F + 2\sigma_1 \|X' - X\|_F$ [3]. Using this and equation (F.45),

[3]By setting $X' = X$ in this bound, we get an estimate (upper bound) for the Lipschitz constant L in DOSLAM as $2\sigma_1^2 + 2\eta$.

we get the following upper bound $\hat{L}$ for $\langle \nabla_A f(A', X') - \nabla_A f(A, X),\, A' - A \rangle$ $+ \langle \nabla_X f(A', X') - \nabla_X f(A, X),\, X' - X \rangle$.

$$\hat{L} = \left(2\sigma_1^2 + 2\eta\right) \|A' - A\|_F^2 + 2\,\|X' - X\|_F^2 + 4\sigma_1 \|A' - A\|_F \|X' - X\|_F$$

Now, $\hat{L}$ can be upper bounded by $L_A \|A' - A\|_F^2 + L_X \|X' - X\|_F^2$. However, there are many possible choices for (L_A, L_X) that work. We pick the choice $L_A = 4\sigma_1^2 + 2\eta$, $L_X = 4$, that minimizes the bound associated with the convergence rate in (9.26). Thus, the estimates satisfy the property that L_X is invariant to scaling of Z, and L_A scales as $\sigma_1^2(Z)$ (since, $\eta = \eta_0\, \sigma_1^2$).

APPENDIX G

RESULTS FOR PROOFS IN CHAPTER 11

G.1 Useful Lemmas

Here, we list three results (based on results in Appendix B as well as [96]) that are used in our convergence proof. The following result is from the Appendix of [96].

Lemma 54. *Consider a bounded vector sequence $\left\{\alpha^k\right\}$ with $\alpha^k \in \mathbb{R}^n$, that converges to α^*. Then, every accumulation point of $\left\{H_s(\alpha^k)\right\}$ belongs to the set $\tilde{H}_s(\alpha^*)$.*

The following result is based on the proof of Lemma 6 of [96].

Lemma 55. *Let $\{W^{q_t}, X^{q_t}\}$ with $W^{q_t} \in \mathbb{R}^{n\times n}$, $X^{q_t} \in \mathbb{R}^{n\times n}$, be a subsequence of $\{W^t, X^t\}$ converging to the accumulation point (W^*, X^*). Let $Z \in \mathbb{R}^{n\times N}$ and $L^{-1} = \left(ZZ^T + \lambda I\right)^{-1/2}$, with $\lambda > 0$. Further, let $Q^{q_t}\Sigma^{q_t}\left(R^{q_t}\right)^T$ denote the full singular value decomposition of $L^{-1}Z\left(X^{q_t}\right)^T$. Let*

$$W^{q_t+1} = \frac{R^{q_t}}{2}\left(\Sigma^{q_t} + \left((\Sigma^{q_t})^2 + 2\lambda I\right)^{\frac{1}{2}}\right)\left(Q^{q_t}\right)^T L^{-1}$$

and suppose that $\{W^{q_t+1}\}$ converges to W^{**}. Then,

$$W^{**} \in \arg\min_{W} \|WZ - X^*\|_F^2 + \lambda\|W\|_F^2 - \lambda\log|\det W| \tag{G.1}$$

The following result is based on the proof of Lemma 9 of [96]. Note that $\phi(X)$ is the barrier function defined in Section 11.4.

Lemma 56. *Given $Z \in \mathbb{R}^{n\times N_1}$, $\lambda > 0$, and $s \geq 0$, consider the function $g : \mathbb{R}^{n\times n} \times \mathbb{R}^{n\times N_1} \mapsto \mathbb{R}$ defined as $g(W, X) = \|WZ - X\|_F^2 + \lambda\|W\|_F^2$ $-\lambda\log|\det W| + \phi(X)$ for $W \in \mathbb{R}^{n\times n}$ and $X \in \mathbb{R}^{n\times N_1}$. Further, let $(\hat{W}, \hat{X})$*

be a pair in $\mathbb{R}^{n\times n}\times\mathbb{R}^{n\times N_1}$ *satisfying*

$$2\hat{W}ZZ^T - 2\hat{X}Z^T + 2\lambda\hat{W} - \lambda\hat{W}^{-T} = 0 \tag{G.2}$$

$$\hat{X}_i \in \tilde{H}_s(\hat{W}Z_i)\,,\ \forall\, 1\leq i\leq N_1 \tag{G.3}$$

Then, the following condition holds at $(\hat{W},\hat{X})$.

$$g(\hat{W}+dW,\hat{X}+\Delta X)\geq g(\hat{W},\hat{X}) \tag{G.4}$$

The condition holds for all sufficiently small $dW\in\mathbb{R}^{n\times n}$ *satisfying* $\|dW\|_F\leq \epsilon'$ *for some* $\epsilon'>0$ *that depends on* $\hat{W}$*, and all* $\Delta X\in\mathbb{R}^{n\times N}$ *in the union of the following regions.*

R1. The half-space tr $\left\{(\hat{W}Z-\hat{X})\Delta X^T\right\}\leq 0$.

R2. The local region defined by
$\|\Delta X\|_\infty < \min_i\left\{\beta_s(\hat{W}Z_i) : \left\|\hat{W}Z_i\right\|_0 > s\right\}$.

Furthermore, if we have $\left\|\hat{W}Z_i\right\|_0\leq s\,\forall\, i$*, then* ΔX *can be arbitrary.*

The following lemma is a slightly modified version of the one in [209]. We only state the minor modifications to the previous proof, for the following lemma to hold.

The lemma implies Lipschitz continuity (and therefore, continuity) of the function $u(B)\triangleq\|By-H_s(By)\|_2^2$ on a bounded set.

Lemma 57. *Given* $c_0>0$*, and* $y\in\mathbb{R}^n$ *satisfying* $\|y\|_2\leq c_0$*, and a constant* $c'>0$*, the function* $u(B)=\|By-H_s(By)\|_2^2$ *is uniformly Lipschitz with respect to* B *on the bounded set* $S\triangleq\{B\in\mathbb{R}^{n\times n}:\|B\|_2\leq c'\}$.

Proof: The proof is identical to that for Lemma 4 in [209], except that the conditions $\|y\|_2=1$ and $\|B\|_2\leq 1$ in [209] are replaced by the conditions $\|y\|_2\leq c_0$ and $\|B\|_2\leq c'$ for the proof here. ■

REFERENCES

[1] M. Elad and M. Aharon, "Image denoising via sparse and redundant representations over learned dictionaries," *IEEE Trans. Image Process.*, vol. 15, no. 12, pp. 3736–3745, 2006.

[2] S. Ravishankar and Y. Bresler, "Learning doubly sparse transforms for images," *IEEE Trans. Image Process.*, vol. 22, no. 12, pp. 4598–4612, 2013.

[3] M. Lustig, D. Donoho, and J. Pauly, "Sparse MRI: The application of compressed sensing for rapid MR imaging," *Magnetic Resonance in Medicine*, vol. 58, no. 6, pp. 1182–1195, 2007.

[4] B. Ning, X. Qu, D. Guo, C. Hu, and Z. Chen, "Magnetic resonance image reconstruction using trained geometric directions in 2d redundant wavelets domain and non-convex optimization," *Magnetic Resonance Imaging*, vol. 31, no. 9, pp. 1611–1622, 2013.

[5] S. Ravishankar and Y. Bresler, "MR image reconstruction from highly undersampled k-space data by dictionary learning," *IEEE Trans. Med. Imag.*, vol. 30, no. 5, pp. 1028–1041, 2011.

[6] S. Ravishankar and Y. Bresler, "Closed-form solutions within sparsifying transform learning," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2013, pp. 5378–5382.

[7] K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian, "Image denoising by sparse 3D transform-domain collaborative filtering," *IEEE Trans. on Image Processing*, vol. 16, no. 8, pp. 2080–2095, 2007.

[8] D. Zoran and Y. Weiss, "From learning models of natural image patches to whole image restoration," in *IEEE International Conference on Computer Vision*, Nov 2011, pp. 479–486.

[9] S. Ravishankar and Y. Bresler, "Learning sparsifying transforms," *IEEE Trans. Signal Process.*, vol. 61, no. 5, pp. 1072–1086, 2013.

[10] A. Hyvärinen and E. Oja, "A fast fixed-point algorithm for independent component analysis," *Neural Comput.*, vol. 9, no. 7, pp. 1483–1492, 1997.

[11] A. Beck and M. Teboulle, “A fast iterative shrinkage-thresholding algorithm for linear inverse problems,” *SIAM Journal on Imaging Sciences*, vol. 2, no. 1, pp. 183–202, 2009.

[12] S. Mallat, *A Wavelet Tour of Signal Processing.* San Diego, CA: Academic Press.

[13] M. W. Marcellin, M. J. Gormish, A. Bilgin, and M. P. Boliek, “An overview of JPEG-2000,” in *Proc. Data Compression Conf.*, 2000, pp. 523–541.

[14] M. Elad, P. Milanfar, and R. Rubinstein, “Analysis versus synthesis in signal priors,” *Inverse Problems*, vol. 23, no. 3, pp. 947–968, 2007.

[15] B. A. Olshausen and D. J. Field, “Emergence of simple-cell receptive field properties by learning a sparse code for natural images,” *Nature*, vol. 381, no. 6583, pp. 607–609, 1996.

[16] K. Engan, S. Aase, and J. Hakon-Husoy, “Method of optimal directions for frame design,” in *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing*, 1999, pp. 2443–2446.

[17] M. Aharon, M. Elad, and A. Bruckstein, “K-SVD: An algorithm for designing overcomplete dictionaries for sparse representation,” *IEEE Transactions on signal processing*, vol. 54, no. 11, pp. 4311–4322, 2006.

[18] M. Yaghoobi, T. Blumensath, and M. Davies, “Dictionary learning for sparse approximations with the majorization method,” *IEEE Transaction on Signal Processing*, vol. 57, no. 6, pp. 2178–2191, 2009.

[19] K. Skretting and K. Engan, “Recursive least squares dictionary learning algorithm,” *IEEE Transactions on Signal Processing*, vol. 58, no. 4, pp. 2121–2130, 2010.

[20] J. Mairal, F. Bach, J. Ponce, and G. Sapiro, “Online learning for matrix factorization and sparse coding,” *J. Mach. Learn. Res.*, vol. 11, pp. 19–60, 2010.

[21] G. Peyré and J. Fadili, “Learning analysis sparsity priors,” in *Proc. Int. Conf. Sampling Theory Appl. (SampTA)*, Singapore, May 2-6, 2011. [Online]. Available: http://hal.archives-ouvertes.fr/hal-00542016/

[22] M. Yaghoobi, S. Nam, R. Gribonval, and M. Davies, “Analysis operator learning for overcomplete cosparse representations,” in *European Signal Processing Conference (EUSIPCO)*, 2011, pp. 1470–1474.

[23] R. Rubinstein and M. Elad, “K-SVD dictionary-learning for analysis sparse models,” in *Proc. SPARS11*, June 2011, p. 73.

[24] B. Ophir, M. Elad, N. Bertin, and M. Plumbley, “Sequential minimal eigenvalues - an approach to analysis dictionary learning,” in *Proc. European Signal Processing Conference (EUSIPCO)*, 2011, pp. 1465–1469.

[25] M. Yaghoobi, S. Nam, R. Gribonval, and M. E. Davies, “Noise aware analysis operator learning for approximately cosparse signals,” in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2012, pp. 5409–5412.

[26] R. Rubinstein, T. Faktor, and M. Elad, “K-SVD dictionary-learning for the analysis sparse model,” in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2012, pp. 5405–5408.

[27] A. M. Bruckstein, D. L. Donoho, and M. Elad, “From sparse solutions of systems of equations to sparse modeling of signals and images,” *SIAM Review*, vol. 51, no. 1, pp. 34–81, 2009.

[28] J. A. Tropp, “Greed is good: Algorithmic results for sparse approximation,” *IEEE Trans. Inform. Theory*, vol. 50, no. 10, pp. 2231–2242, 2004.

[29] E. Candès and T. Tao, “Decoding by linear programming,” *IEEE Trans. on Information Theory*, vol. 51, no. 12, pp. 4203–4215, 2005.

[30] E. Candès, J. Romberg, and T. Tao, “Robust uncertainty principles: exact signal reconstruction from highly incomplete frequency information,” *IEEE Trans. Information Theory*, vol. 52, no. 2, pp. 489–509, 2006.

[31] D. Donoho, “Compressed sensing,” *IEEE Trans. Information Theory*, vol. 52, no. 4, pp. 1289–1306, 2006.

[32] B. K. Natarajan, “Sparse approximate solutions to linear systems,” *SIAM J. Comput.*, vol. 24, no. 2, pp. 227–234, Apr. 1995.

[33] G. Davis, S. Mallat, and M. Avellaneda, “Adaptive greedy approximations,” *Journal of Constructive Approximation*, vol. 13, no. 1, pp. 57–98, 1997.

[34] Y. Pati, R. Rezaiifar, and P. Krishnaprasad, “Orthogonal matching pursuit : recursive function approximation with applications to wavelet decomposition,” in *Asilomar Conf. on Signals, Systems and Comput.*, 1993, pp. 40–44 vol.1.

[35] S. G. Mallat and Z. Zhang, “Matching pursuits with time-frequency dictionaries,” *IEEE Transactions on Signal Processing*, vol. 41, no. 12, pp. 3397–3415, 1993.

[36] W. Dai and O. Milenkovic, "Subspace pursuit for compressive sensing signal reconstruction," *IEEE Trans. Information Theory*, vol. 55, no. 5, pp. 2230–2249, 2009.

[37] S. S. Chen, D. L. Donoho, and M. A. Saunders, "Atomic decomposition by basis pursuit," *SIAM J. Sci. Comput.*, vol. 20, no. 1, pp. 33–61, 1998.

[38] I. F. Gorodnitsky, J. George, and B. D. Rao, "Neuromagnetic source imaging with FOCUSS: A recursive weighted minimum norm algorithm," *Electrocephalography and Clinical Neurophysiology*, vol. 95, pp. 231–251, 1995.

[39] B. Efron, T. Hastie, I. Johnstone, and R. Tibshirani, "Least angle regression," *Annals of Statistics*, vol. 32, pp. 407–499, 2004.

[40] D. L. Donoho, "For most large underdetermined systems of linear equations the minimal l1-norm solution is also the sparsest solution," *Comm. Pure Appl. Math*, vol. 59, pp. 797–829, 2004.

[41] G. Harikumar and Y. Bresler, "A new algorithm for computing sparse solutions to linear inverse problems," in *ICASSP*, may 1996, pp. 1331–1334.

[42] G. Harikumar, "Blind image deconvolution from multiple blurs, and sparse approximation," Ph.D. dissertation, University of Illinois at Urbana-Champaign, mar 1997, Yoram Bresler, adviser.

[43] Y. Bresler, C. Couvreur, and G. Harikumar, "Fast optimal and suboptimal algorithms for sparse solutions to linear inverse problems," in *Proc. IEEE Int. Conf. Acoust. Speech, Sig. Proc.*, vol. 3, apr 1998, pp. 1877–1880.

[44] R. Chartrand, "Exact reconstruction of sparse signals via nonconvex minimization," *Signal Processing Letters, IEEE*, vol. 14, no. 10, pp. 707–710, 2007.

[45] E. J. Candès and D. L. Donoho, "Ridgelets: A key to higher-dimensional intermittency?" *Phil. Trans. R. Soc. Lond. A*, vol. 357, no. 1760, pp. 2495–2509, 1999.

[46] M. N. Do and M. Vetterli, "The contourlet transform: an efficient directional multiresolution image representation," *IEEE Trans. Image Process.*, vol. 14, no. 12, pp. 2091–2106, 2005.

[47] E. J. Candès and D. L. Donoho, "Curvelets - a surprisingly effective nonadaptive representation for objects with edges," in *Curves and Surfaces.* Vanderbilt University Press, 1999, pp. 105–120.

[48] R. Gribonval and K. Schnass, "Dictionary identification–sparse matrix-factorization via l_1 -minimization," *IEEE Trans. Inform. Theory*, vol. 56, no. 7, pp. 3523–3539, 2010.

[49] S. K. Sahoo and A. Makur, "Dictionary training for sparse representation as generalization of k-means clustering," *IEEE Signal Processing Letters*, vol. 20, no. 6, pp. 587–590, June 2013.

[50] L. N. Smith and M. Elad, "Improving dictionary learning: Multiple dictionary updates and coefficient reuse," *IEEE Signal Processing Letters*, vol. 20, no. 1, pp. 79–82, Jan 2013.

[51] M. Sadeghi, M. Babaie-Zadeh, and C. Jutten, "Dictionary learning for sparse representation: A novel approach," *IEEE Signal Processing Letters*, vol. 20, no. 12, pp. 1195–1198, Dec 2013.

[52] Y. Chen and X. Ye, "A novel method and fast algorithm for MR image reconstruction with significantly under-sampled data," *Inverse Problems and Imaging*, vol. 4, no. 2, pp. 223–240, 2010.

[53] R. Rubinstein, A. M. Bruckstein, and M. Elad, "Dictionaries for sparse representation modeling," *Proceedings of the IEEE*, vol. 98, no. 6, pp. 1045–1057, 2010.

[54] Q. Geng, H. Wang, and J. Wright, "On the local correctness of ℓ^1-minimization for dictionary learning," 2011, preprint: http://arxiv.org/abs/1101.5672.

[55] D. A. Spielman, H. Wang, and J. Wright, "Exact recovery of sparsely-used dictionaries," 2012, preprint: http://arxiv.org/abs/1206.5882.

[56] J. Mairal, M. Elad, and G. Sapiro, "Sparse representation for color image restoration," *IEEE Trans. on Image Processing*, vol. 17, no. 1, pp. 53–69, 2008.

[57] M. Protter and M. Elad, "Image sequence denoising via sparse and redundant representations," *IEEE Trans. on Image Processing*, vol. 18, no. 1, pp. 27–36, 2009.

[58] M. Aharon and M. Elad, "Sparse and redundant modeling of image content using an image-signature-dictionary," *SIAM Journal on Imaging Sciences*, vol. 1, no. 3, pp. 228–247, 2008.

[59] J. Mairal, G. Sapiro, and M. Elad, "Learning multiscale sparse representations for image and video restoration," *SIAM Multiscale Modeling and Simulation*, vol. 7, no. 1, pp. 214–241, 2008.

[60] R. Rubinstein, M. Zibulevsky, and M. Elad, “Double sparsity: Learning sparse dictionaries for sparse signal approximation,” *IEEE Transactions on Signal Processing*, vol. 58, no. 3, pp. 1553–1564, 2010.

[61] G. Yu, G. Sapiro, and S. Mallat, “Image modeling and enhancement via structured sparse model selection,” in *Proc. IEEE International Conference on Image Processing (ICIP)*, 2010, pp. 1641–1644.

[62] I. Ramirez, P. Sprechmann, and G. Sapiro, “Classification and clustering via dictionary learning with structured incoherence and shared features,” in *Proc. IEEE International Conference on Computer Vision and Pattern Recognition (CVPR) 2010*, 2010, pp. 3501–3508.

[63] H. Y. Liao and G. Sapiro, “Sparse representations for limited data tomography,” in *Proc. IEEE International Symposium on Biomedical Imaging (ISBI)*, 2008, pp. 1375–1378.

[64] S. Ravishankar and Y. Bresler, “Multiscale dictionary learning for MRI,” in *Proc. ISMRM*, 2011, p. 2830.

[65] A. Chambolle, “An algorithm for total variation minimization and applications,” *J. Math. Imaging Vis.*, vol. 20, no. 1-2, pp. 89–97, 2004.

[66] S. Nam, M. E. Davies, M. Elad, and R. Gribonval, “Cosparse analysis modeling - uniqueness and algorithms,” in *ICASSP*, 2011, pp. 5804–5807.

[67] E. J. Candès, Y. C. Eldar, D. Needell, and P. Randall, “Compressed sensing with coherent and redundant dictionaries,” *Applied and Computational Harmonic Analysis*, vol. 31, no. 1, pp. 59–73, 2011.

[68] Y. Liu, M. Tiebin, and S. Li, “Compressed sensing with general frames via optimal-dual-based ℓ_1-analysis,” *IEEE Transactions on Information Theory*, vol. 58, no. 7, pp. 4201–4214, July 2012.

[69] R. Giryes, S. Nam, M. Elad, R. Gribonval, and M. Davies, “Greedy-like algorithms for the cosparse analysis model,” *Linear Algebra and its Applications*, vol. 441, pp. 22–60, 2014, special Issue on Sparse Approximate Solution of Linear Systems.

[70] M. Yaghoobi, S. Nam, R. Gribonval, and M. E. Davies, “Constrained overcomplete analysis operator learning for cosparse signal modelling,” *IEEE Trans. Signal Process.*, vol. 61, no. 9, pp. 2341–2355, 2013.

[71] Y. Chen, T. Pock, and H. Bischof, “Learning ℓ_1-based analysis and synthesis sparsity priors using bi-level optimization,” in *Proc. Workshop on Analysis Operator Learning vs. Dictionary Learning, NIPS*, 2012. [Online]. Available: http://arxiv.org/abs/1401.4105

[72] S. Hawe, M. Kleinsteuber, and K. Diepold, “Analysis operator learning and its application to image reconstruction,” *IEEE Transactions on Image Processing*, vol. 22, no. 6, pp. 2138–2150, June 2013.

[73] M. Elad, “Michael Elad personal page,” http://www.cs.technion.ac.il/~elad/talks/, [Online; accessed 2014].

[74] E. J. Candès and J. Romberg, “Signal recovery from random projections,” in *SPIE International Symposium on Electronic Imaging: Computational Imaging III*, 2005, pp. 76–86.

[75] W. K. Pratt, J. Kane, and H. C. Andrews, “Hadamard transform image coding,” *Proc. IEEE*, vol. 57, no. 1, pp. 58–68, 1969.

[76] J. B. Allen and L. R. Rabiner, “A unified approach to short-time fourier analysis and synthesis,” *Proc. IEEE*, vol. 65, no. 11, pp. 1558–1564, 1977.

[77] C. Lemaréchal and C. Sagastizábal, “Practical aspects of the moreau–yosida regularization: Theoretical preliminaries,” *SIAM J. on Optimization*, vol. 7, no. 2, pp. 367–385, 1997.

[78] M. Lustig, J. M. Santos, D. L. Donoho, and J. M. Pauly, “k-t SPARSE: High frame rate dynamic MRI exploiting spatio-temporal sparsity,” in *Proc. ISMRM*, 2006, p. 2420.

[79] S. Ravishankar and Y. Bresler, “Learning sparsifying transforms for signal and image processing,” in *SIAM Conference on Imaging Science*, 2012, p. 51.

[80] S. Ravishankar and Y. Bresler, “Learning sparsifying transforms for image processing,” in *IEEE Int. Conf. Image Process.*, 2012, pp. 681–684.

[81] C. Wang, D. Sun, and K.-C. Toh, “Solving log-determinant optimization problems by a newton-CG primal proximal point algorithm,” *SIAM J. Optim*, vol. 20, no. 6, pp. 2994–3013, 2010.

[82] M. Zibulevsky and B. A. Pearlmutter, “Blind source separation by sparse decomposition in a signal dictionary,” *Neural Computation*, vol. 13, no. 4, pp. 863–882, 2001.

[83] R. Pytlak, *Conjugate Gradient Algorithms in Nonconvex Optimization.* Heidelberg, Germany: Springer-Verlag, 2009.

[84] J. Dattorro, *Convex Optimization & Euclidean Distance Geometry.* Palo Alto, California: Meboo Publishing USA, 2005.

[85] J. A. Fill and S. Janson, “Quicksort asymptotics,” *Journal of Algorithms*, vol. 44, no. 1, pp. 4–28, 2002.

[86] M. Elad, “Michael Elad personal page,” http://www.cs.technion.ac.il/~elad/Various/KSVD_Matlab_ToolBox.zip, [Online; accessed 2014].

[87] S. Ravishankar and Y. Bresler, “Learning sparsifying transforms for image processing,” in *IEEE Int. Conf. Image Process.*, 2012, pp. 681–684.

[88] S. Ravishankar and Y. Bresler, “Learning doubly sparse transforms for image representation,” in *IEEE Int. Conf. Image Process.*, 2012, pp. 685–688.

[89] B. Ophir, M. Lustig, and M. Elad, “Multi-scale dictionary learning using wavelets,” *IEEE Journal of Selected Topics in Signal Processing*, vol. 5, no. 5, pp. 1014–1024, 2011.

[90] J. Portilla, “Image restoration through l0 analysis-based sparse optimization in tight frames,” in *IEEE International Conference on Image Processing (ICIP)*, 2009, pp. 3909–3912.

[91] “CVG-UGR-Image database,” http://decsai.ugr.es/cvg/dbimagenes/index.php, [Online; accessed 29-April-2012].

[92] T. A. Davis, *Direct Methods for Sparse Linear Systems.* Philadelphia, PA: SIAM, 2006.

[93] Y. Saad, *Iterative Methods for Sparse Linear Systems.* Philadelphia, PA: SIAM, 2003.

[94] R. Rubinstein, http://www.cs.technion.ac.il/~ronrubin/Software/ksvdsbox11.zip, [Online; accessed 09-October-2012].

[95] S. Ravishankar and Y. Bresler, “Closed-form optimal updates in transform learning,” in *Signal Processing with Adaptive Sparse Structured Representations (SPARS) workshop*, Lausanne, Switzerland, July 2013.

[96] S. Ravishankar and Y. Bresler, “ℓ_0 sparsifying transform learning with efficient optimal updates and convergence guarantees,” *IEEE Trans. Signal Process.*, 2014, https://uofi.box.com/s/oq8xokc7ozgv3yl47ytc. Accepted with Minor Revisions.

[97] L. Pfister, “Tomographic reconstruction with adaptive sparsifying transforms,” M.S. thesis, University of Illinois at Urbana-Champaign, Aug. 2013.

[98] L. Pfister and Y. Bresler, "Model-based iterative tomographic reconstruction with adaptive sparsifying transforms," in *SPIE International Symposium on Electronic Imaging: Computational Imaging XII*, 2014, to appear.

[99] L. Mirsky, "On the trace of matrix products," *Mathematische Nachrichten*, vol. 20, no. 3-6, pp. 171–174, 1959.

[100] P. Tseng, "Convergence of a block coordinate descent method for nondifferentiable minimization," *J. Optim. Theory Appl.*, vol. 109, no. 3, pp. 475–494, 2001.

[101] R. A. Horn and C. R. Johnson, *Matrix Analysis.* Cambridge, United Kingdom: Cambridge University Press, 1985.

[102] T. Blumensath and M. E. Davies, "Iterative hard thresholding for compressed sensing," *Applied and Computational Harmonic Analysis*, vol. 27, no. 3, pp. 265–274, 2009.

[103] T. Blumensath and M. E. Davies, "Iterative thresholding for sparse approximations," *Journal of Fourier Analysis and Applications*, vol. 14, no. 5-6, pp. 629–654, 2008.

[104] S. Ravishankar and Y. Bresler, "Sparsifying transform learning for compressed sensing MRI," in *Proc. IEEE Int. Symp. Biomed. Imag.*, 2013, pp. 17–20.

[105] A. J. Bell and T. J. Sejnowski, "An information-maximization approach to blind separation and blind deconvolution," *Neural Comput.*, vol. 7, no. 6, pp. 1129–1159, 1995.

[106] A. J. Bell and T. J. Sejnowski, "The "independent components of natural scenes are edge filters," *Vision Research*, vol. 37, no. 23, pp. 3327–3338, 1997.

[107] A. Hyvärinen, J. Karhunen, and E. Oja, *Independent Component Analysis.* New York, NY: Wiley-Interscience, 2001.

[108] J.-F. Cardoso, "High-order contrasts for independent component analysis," *Neural Comput.*, vol. 11, no. 1, pp. 157–192, 1999.

[109] A. J. Ferreira and M. A. T. Figueiredo, "Class-adapted image compression using independent component analysis," in *Proceedings. 2003 International Conference on Image Processing*, vol. 1, 2003, pp. I–625–8 vol.1.

[110] A. Hyvärinen, "The fastica matlab package," http://research.ics.aalto.fi/ica/fastica/code/FastICA_2.5.zip, [Online; accessed August-2014].

[111] J.-F. Cardoso, “Jade for real-valued data,” http://bsp.teithe.gr/members/downloads/Jade, [Online; accessed August-2014].

[112] P. Feng and Y. Bresler, “Spectrum-blind minimum-rate sampling and reconstruction of multiband signals,” in *ICASSP*, vol. 3, may 1996, pp. 1689–1692.

[113] Y. Bresler and P. Feng, “Spectrum-blind minimum-rate sampling and reconstruction of 2-D multiband signals,” in *Proc. 3rd IEEE Int. Conf. on Image Processing, ICIP’96*, sep 1996, pp. 701–704.

[114] P. Feng, “Universal spectrum blind minimum rate sampling and reconstruction of multiband signals,” Ph.D. dissertation, University of Illinois at Urbana-Champaign, mar 1997, Yoram Bresler, adviser.

[115] R. Venkataramani and Y. Bresler, “Further results on spectrum blind sampling of 2D signals,” in *Proc. IEEE Int. Conf. Image Proc., ICIP*, vol. 2, Oct. 1998, pp. 752–756.

[116] Y. Bresler, M. Gastpar, and R. Venkataramani, “Image compression on-the-fly by universal sampling in Fourier imaging systems,” in *Proc. 1999 IEEE Information Theory Workshop on Detection, Estimation, Classification, and Imaging*, feb 1999, p. 48.

[117] M. Gastpar and Y. Bresler, “On the necessary density for spectrum-blind nonuniform sampling subject to quantization,” in *ICASSP*, vol. 1, jun 2000, pp. 348–351.

[118] J. C. Ye, Y. Bresler, and P. Moulin, “A self-referencing level-set method for image reconstruction from sparse Fourier samples,” *Int. J. Computer Vision*, vol. 50, no. 3, pp. 253–270, dec 2002.

[119] Y. Bresler, “Spectrum-blind sampling and compressive sensing for continuous-index signals,” in *2008 Information Theory and Applications Workshop Conference*, 2008, pp. 547–554.

[120] D. L. Donoho, M. Elad, and V. N. Temlyakov, “Stable recovery of sparse overcomplete representations in the presence of noise,” *IEEE Trans. Inform. Theory*, vol. 52, no. 1, pp. 6–18, 2006.

[121] R. Chartrand, “Fast algorithms for nonconvex compressive sensing: MRI reconstruction from very few data,” in *Proc. IEEE International Symposium on Biomedical Imaging (ISBI)*, 2009, pp. 262–265.

[122] J. Trzasko and A. Manduca, “Highly undersampled magnetic resonance image reconstruction via homotopic l_0-minimization,” *IEEE Trans. Med. Imaging*, vol. 28, no. 1, pp. 106–121, 2009.

[123] Y. Kim, M. S. Nadar, and A. Bilgin, “Wavelet-based compressed sensing using gaussian scale mixtures,” in *Proc. ISMRM*, 2010, p. 4856.

[124] C. Qiu, W. Lu, and N. Vaswani, “Real-time dynamic MR image reconstruction using kalman filtered compressed sensing,” in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing*, 2009, pp. 393–396.

[125] X. Qu, D. Guo, B. Ning, Y. Hou, Y. Lin, S. Cai, and Z. Chen, “Undersampled MRI reconstruction with patch-based directional wavelets,” *Magnetic Resonance Imaging*, vol. 30, no. 7, pp. 964–977, 2012.

[126] G. H. Chen, J. Tang, and S. Leng, “Prior image constrained compressed sensing (piccs): A method to accurately reconstruct dynamic CT images from highly undersampled projection data sets,” *Med. Phys.*, vol. 35, no. 2, pp. 660–663, 2008.

[127] K. Choi, J. Wang, L. Zhu, T.-S. Suh, S. Boyd, and L. Xing, “Compressed sensing based cone-beam computed tomography reconstruction with a first-order method,” *Med. Phys.*, vol. 37, no. 9, pp. 5113–5125, 2010.

[128] X. Li and S. Luo, “A compressed sensing-based iterative algorithm for ct reconstruction and its possible application to phase contrast imaging,” *BioMedical Engineering OnLine*, vol. 10, no. 1, p. 73, 2011.

[129] S. Valiollahzadeh, T. Chang, J. W. Clark, and O. R. Mawlawi, “Image recovery in pet scanners with partial detector rings using compressive sensing,” in *IEEE Nuclear Science Symposium and Medical Imaging Conference (NSS/MIC)*, Oct 2012, pp. 3036–3039.

[130] K. Malczewski, “Pet image reconstruction using compressed sensing,” in *Signal Processing: Algorithms, Architectures, Arrangements, and Applications (SPA), 2013*, Sept 2013, pp. 176–181.

[131] S. Gleichman and Y. C. Eldar, “Blind compressed sensing,” *IEEE Transactions on Information Theory*, vol. 57, no. 10, pp. 6958–6975, 2011.

[132] S. G. Lingala and M. Jacob, “Blind compressive sensing dynamic mri,” *IEEE Transactions on Medical Imaging*, vol. 32, no. 6, pp. 1132–1145, 2013.

[133] Y. Wang, Y. Zhou, and L. Ying, “Undersampled dynamic magnetic resonance imaging using patch-based spatiotemporal dictionaries,” in *2013 IEEE 10th International Symposium on Biomedical Imaging (ISBI)*, April 2013, pp. 294–297.

[134] B. Wen, S. Ravishankar, and Y. Bresler, “Structured overcomplete sparsifying transform learning with convergence guarantees and applications,” *International Journal of Computer Vision*, 2014, to appear.

[135] K. P. Pruessmann, “Encoding and reconstruction in parallel MRI,” *NMR in Biomedicine*, vol. 19, no. 3, pp. 288–299, 2006.

[136] L. Eldèn, “Solving quadratically constrained least squares problems using a differential-geometric approach,” *BIT Numerical Mathematics*, vol. 42, no. 2, pp. 323–335, 2002.

[137] P. Tseng, “Convergence of a block coordinate descent method for non-differentiable minimization,” *J. Optim. Theory Appl.*, vol. 109, no. 3, pp. 475–494, 2001.

[138] Y. Xu and W. Yin, “A block coordinate descent method for regularized multiconvex optimization with applications to nonnegative tensor factorization and completion,” *SIAM Journal on Imaging Sciences*, vol. 6, no. 3, pp. 1758–1789, 2013.

[139] E. Chouzenoux, J.-C. Pesquet, and A. Repetti, “A block coordinate variable metric forward-backward algorithm,” 2014, preprint: http://www.optimization-online.org/DB_HTML/2013/12/4178.html.

[140] R. T. Rockafellar and R. J.-B. Wets, *Variational Analysis.* Heidelberg, Germany: Springer-Verlag, 1998.

[141] B. S. Mordukhovich, *Variational Analysis and Generalized Differentiation. Vol. I: Basic theory.* Heidelberg, Germany: Springer-Verlag, 2006.

[142] M. Lustig, “Michael Lustig home page,” http://www.eecs.berkeley.edu/~mlustig/Software.html, [Online; accessed October, 2014].

[143] X. Qu, “PBDWS code,” http://www.quxiaobo.org/project/CS_MRI_PBDWS/Demo_PBDWS_SparseMRI.zip, [Online; accessed September, 2014].

[144] S. Ravishankar and Y. Bresler, “DLMRI - Lab: Dictionary learning MRI software,” http://www.ifp.illinois.edu/~yoram/DLMRI-Lab/DLMRI.html, [Online; accessed October, 2014].

[145] S. Ravishankar and Y. Bresler, “Adaptive sampling design for compressed sensing MRI,” in *Conf. Proc. IEEE Eng. Med. Biol. Soc.*, 2011, pp. 3751–3755.

[146] H. Wang, D. Liang, and L. Ying, “Pseudo 2D random sampling for compressed sensing mri,” in *Conf. Proc. IEEE Eng. Med. Biol. Soc.*, 2009, pp. 2672–2675.

[147] M. Seeger, H. Nickisch, R. Pohmann, and B. Schölkopf, "Optimization of k-space trajectories for compressed sensing by bayesian experimental design," *Magn. Reson. Med.*, vol. 63, no. 1, pp. 116–126, 2010.

[148] S. Ravishankar, B. Wen, and Y. Bresler, "Online sparsifying transform learning - part I: Algorithms," *IEEE Journal of Selected Topics in Signal Process.*, 2014, https://uofi.box.com/s/3kwdao7y9tbeohrzdqh5. Accepted for publication.

[149] S. Ravishankar, B. Wen, and Y. Bresler, "Online sparsifying transform learning for signal processing," in *IEEE Global Conference on Signal and Information Processing (GlobalSIP)*, 2014, to appear.

[150] S. Ravishankar and Y. Bresler, "Online sparsifying transform learning - part II: Convergence analysis," *IEEE Journal of Selected Topics in Signal Process.*, 2014, accepted.

[151] C. Lu, J. Shi, and J. Jia, "Online robust dictionary learning," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2013, pp. 415–422.

[152] J. Xing, J. Gao, B. Li, W. Hu, and S. Yan, "Robust object tracking with online multi-lifespan dictionary learning," in *IEEE International Conference on Computer Vision (ICCV)*, Dec 2013, pp. 665–672.

[153] G. Zhang, Z. Jiang, and L. S. Davis, "Online semi-supervised discriminative dictionary learning for sparse representation," in *Computer Vision ACCV 2012*, 2013, pp. 259–273.

[154] L. Wang, K. Lu, P. Liu, R. Ranjan, and L. Chen, "Ik-svd: Dictionary learning for spatial big data via incremental atom update," *Computing in Science Engineering*, vol. 16, no. 4, pp. 41–52, July 2014.

[155] S. Mukherjee and C. S. Seelamantula, "A split-and-merge dictionary learning algorithm for sparse representation," 2014, preprint: http://arxiv.org/abs/1403.4781.

[156] P. Stange, "On the efficient update of the singular value decomposition," *PAMM*, vol. 8, no. 1, pp. 10 827–10 828, 2008.

[157] D. L. Donoho and I. M. Johnstone, "Ideal spatial adaptation by wavelet shrinkage," *Biometrika*, vol. 81, no. 3, pp. 425–455, 1994.

[158] "The USC-SIPI Image Database," http://sipi.usc.edu/database/database.php?volume=misc, [Online; accessed July-2014].

[159] Y. Z. Tsypkin, *Adaptation and Learning in automatic systems.* New York, NY: Academic Press.

[160] Y. Z. Tsypkin, *Foundations of the theory of learning systems.* New York, NY: Academic Press, 1973.

[161] L. Bottou, "Online algorithms and stochastic approximations," in *Online Learning and Neural Networks*, D. Saad, Ed. Cambridge, UK: Cambridge University Press, 1998.

[162] L. Bottou and O. Bousquet, "The tradeoffs of large scale learning," in *Advances in Neural Information Processing Systems*, vol. 20. MIT Press, 2008, pp. 161–168.

[163] A. W. van der Vaart, *Asymptotic Statistics.* Cambridge, UK: Cambridge University Press, 2000.

[164] S. Ravishankar and Y. Bresler, "Doubly sparse transform learning with convergence guarantees," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, May 2014, pp. 5262–5266.

[165] A. Chambolle, R. A. D. Vore, N.-Y. Lee, and B. J. Lucier, "Nonlinear wavelet image processing: variational problems, compression, and noise removal through wavelet shrinkage," *IEEE Transactions on Image Processing*, vol. 7, no. 3, pp. 319–335, 1998.

[166] J. M. Bioucas-Dias and M. A. T. Figueiredo, "A new twist: Two-step iterative shrinkage/thresholding algorithms for image restoration," *IEEE Transactions on Image Processing*, vol. 16, no. 12, pp. 2992–3004, 2007.

[167] S. Becker, J. Bobin, and E. Candès, "Nesta: A fast and accurate first-order method for sparse recovery," *SIAM Journal on Imaging Sciences*, vol. 4, no. 1, pp. 1–39, 2011.

[168] S. Boyd and L. Vandenberghe, *Convex Optimization.* Cambridge, UK: Cambridge University Press, 2004.

[169] D. P. Bertsekas, *Nonlinear Programming.* Belmont, Massachusetts: Athena Scientific, 1999.

[170] A. Beck and M. Teboulle, "Fast gradient-based algorithms for constrained total variation image denoising and deblurring problems," *IEEE Transactions on Image Processing*, vol. 18, no. 11, pp. 2419–2434, 2009.

[171] S. Foucart, "Hard thresholding pursuit: An algorithm for compressive sensing," *SIAM J. Numer. Anal.*, vol. 49, no. 6, pp. 2543–2563, 2011.

[172] S. Ravishankar and Y. Bresler, "Learning overcomplete sparsifying transforms for signal processing," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, May 2013, pp. 3088–3092.

[173] S. Ravishankar and Y. Bresler, "Learning overcomplete signal sparsifying transforms," in *Signal Processing with Adaptive Sparse Structured Representations (SPARS) workshop*, Lausanne, Switzerland, July 2013.

[174] M. Elad, "Optimized projections for compressed-sensing," *IEEE Trans. on Signal Processing*, vol. 55, no. 12, pp. 5695–5702, 2007.

[175] K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian, "Image denoising by sparse 3-d transform-domain collaborative filtering," *IEEE Trans. Image Process.*, vol. 16, no. 8, pp. 2080–2095, 2007.

[176] B. Wen, S. Ravishankar, and Y. Bresler, "Learning overcomplete sparsifying transforms with block cosparsity," in *Proc. IEEE International Conference on Image Processing (ICIP)*, 2014, to appear.

[177] T. Peleg and M. Elad, "A statistical prediction model based on sparse representations for single image super-resolution," *IEEE Transactions on Image Processing*, vol. 23, no. 6, pp. 2569–2582, 2014.

[178] S. Wang, L. Zhang, Y. Liang, and Q. Pan, "Semi-coupled dictionary learning with applications to image super-resolution and photo-sketch synthesis," in *IEEE Conference on Computer Vision and Pattern Recognition*, June 2012, pp. 2216–2223.

[179] I. Ramirez, P. Sprechmann, and G. Sapiro, "Classification and clustering via dictionary learning with structured incoherence and shared features," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR).* IEEE, 2010, pp. 3501–3508.

[180] S. Kong and D. Wang, "A dictionary learning approach for classification: Separating the particularity and the commonality," in *Proceedings of the 12th European Conference on Computer Vision*, 2012, pp. 186–199.

[181] Y.-C. Chen, C. S. Sastry, V. M. Patel, P. J. Phillips, and R. Chellappa, "Rotation invariant simultaneous clustering and dictionary learning," in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2012, pp. 1053–1056.

[182] P. Sprechmann, A. Bronstein, and G. Sapiro, "Learning efficient structured sparse models," in *Proceedings of the 29th International Conference on Machine Learning*, vol. 1, 2012, pp. 615–622.

[183] P. Sprechmann, A. M. Bronstein, and G. Sapiro, "Learning efficient sparse and low rank models," 2012, preprint: http://arxiv.org/abs/1212.3631.

[184] L. Zelnik-Manor, K. Rosenblum, and Y. C. Eldar, "Dictionary optimization for block-sparse representations," *IEEE Transactions on Signal Processing*, vol. 60, no. 5, pp. 2386–2395, May 2012.

[185] Y.-T. Chi, M. Ali, A. Rajwade, and J. Ho, "Block and group regularized sparse modeling for dictionary learning," in *CVPR*, 2013, pp. 377–382.

[186] D. L. Donoho and M. Elad, "Optimally sparse representation in general (nonorthogonal) dictionaries via 1 minimization," *Proceedings of the National Academy of Sciences*, vol. 100, no. 5, pp. 2197–2202, 2003.

[187] G. Yu, G. Sapiro, and S. Mallat, "Solving inverse problems with piecewise linear estimators: From gaussian mixture models to structured sparsity," *IEEE Transactions on Image Processing*, vol. 21, no. 5, pp. 2481–2499, 2012.

[188] D. A. Spielman, H. Wang, and J. Wright, "Exact recovery of sparsely-used dictionaries," in *Proceedings of the 25th Annual Conference on Learning Theory*, 2012, pp. 37.1–37.18.

[189] A. Agarwal, A. Anandkumar, P. Jain, P. Netrapalli, and R. Tandon, "Learning sparsely used overcomplete dictionaries via alternating minimization," 2013, preprint: http://arxiv.org/abs/1310.7991.

[190] S. Arora, R. Ge, and A. Moitra, "New algorithms for learning incoherent and overcomplete dictionaries," 2013, preprint: http://arxiv.org/pdf/1308.6273v5.pdf.

[191] Y. Xu and W. Yin, "A fast patch-dictionary method for whole-image recovery," 2013, UCLA CAM report 13-38. [Online]. Available: ftp://ftp.math.ucla.edu/pub/camreport/cam13-38.pdf

[192] C. Bao, H. Ji, Y. Quan, and Z. Shen, "ℓ_0 norm based dictionary learning by proximal methods with global convergence," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2014, to appear. Online: http://www.math.nus.edu.sg/~matzuows/BJQS.pdf.

[193] A. Agarwal, A. Anandkumar, P. Jain, P. Netrapalli, and R. Tandon, "Learning sparsely used overcomplete dictionaries," *Journal of Machine Learning Research*, vol. 35, pp. 1–15, 2014.

[194] J. Mairal, F. Bach, J. Ponce, G. Sapiro, and A. Zisserman, "Non-local sparse models for image restoration," in *IEEE International Conference on Computer Vision*, Sept 2009, pp. 2272–2279.

[195] Y. Weiss, "Yair Weiss home page," http://www.cs.huji.ac.il/~daniez/epllcode.zip, [Online; accessed 2014].

[196] K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian, "BM3D web page," http://www.cs.tut.fi/~foi/GCF-BM3D/, [Online; accessed 2014].

[197] P. Brodatz, *Textures: a Photographic Album for Artists and Designers*. New York, NY: Dover, 1966.

[198] D.-C. He and A. Safia, "Multiband Texture Database," http://multibandtexture.recherche.usherbrooke.ca/original_brodatz.html, [Online; accessed 2014].

[199] P. Schönemann, "A generalized solution of the orthogonal procrustes problem," *Psychometrika*, vol. 31, no. 1, pp. 1–10, 1966.

[200] M. Zibulevsky, "Blind source separation with relative newton method," in *4th International Symposium on Independent Component Analysis and Blind Signal Separation*, 2003, pp. 897–902.

[201] H. Attouch, J. Bolte, P. Redont, and A. Soubeyran, "Proximal alternating minimization and projection methods for nonconvex problems: An approach based on the kurdyka-łojasiewicz inequality," *Math. Oper. Res.*, vol. 35, no. 2, pp. 438–457, May 2010.

[202] N. G. Markley, *Principles of Differential Equations*. Hoboken, New Jersey: John Wiley & Sons, Inc., 2004.

[203] G. W. Stewart, "Perturbation theory for the singular value decomposition," in *SVD and Signal Processing, II: Algorithms, Analysis and Applications*. Elsevier, 1990, pp. 99–109.

[204] F. M. Dopico, "A note on $\sin\theta$ theorems for singular subspace variations," *BIT Numerical Mathematics*, vol. 40, no. 2, pp. 395–403, 2000.

[205] L. Hogben, *Handbook of Linear Algebra*. Boca Raton, FL: CRC Press, 2006.

[206] D. L. Fisk, "Quasi-Martingales," *Transactions of the American Mathematical Society*, vol. 120, pp. 369–389, 1965.

[207] J. M. Danskin, *The Theory of Max-Min and Its Applications to Weapons Allocation Problems*. Heidelberg, Germany: Springer, 1967.

[208] J. F. Bonnans and A. Shapiro, "Optimization problems with perturbations: A guided tour," *SIAM Review*, vol. 40, no. 2, pp. 202–227, 1998.

[209] S. Ravishankar and Y. Bresler, "Online sparsifying transform learning: Part II: Convergence analysis," *IEEE Journal of Selected Topics in Signal Process.*, 2014, https://uofi.box.com/s/e691o7vjiuv6rfwvwo6d. Accepted for publication.