Optimization for Probabilistic Machine Learning

Ghazal Fazelnia

Submitted in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy
in the Graduate School of Arts and Sciences

COLUMBIA UNIVERSITY

2019

ABSTRACT

Optimization for Probabilistic Machine Learning

by

Ghazal Fazelnia

We have access to great variety of datasets more than any time in the history. Everyday, more data is collected from various natural resources and digital platforms. Great advances in the area of machine learning research in the past few decades have relied strongly on availability of these datasets. However, analyzing them imposes significant challenges that are mainly due to two factors. First, the datasets have complex structures with hidden interdependencies. Second, most of the valuable datasets are high dimensional and are largely scaled. The main goal of a machine learning framework is to design a model that is a valid representative of the observations and develop a learning algorithm to make inference about unobserved or latent data based on the observations. Discovering hidden patterns and inferring latent characteristics in such datasets is one of the greatest challenges in the area of machine learning research. In this dissertation, I will investigate some of the challenges in modeling and algorithm design, and present my research results on how to overcome these obstacles.

Analyzing data generally involves two main stages. The first stage is designing a model that is flexible enough to capture complex variation and latent structures in data and is robust enough to generalize well to the unseen data. Designing an expressive and interpretable model is one of crucial objectives in this stage. The second stage involves training learning algorithm on the observed data and measuring the accuracy of model and learning algorithm. This stage usually involves an optimization problem whose objective is to tune the model to the training data and learn the model parameters. Finding global optimal or sufficiently good local optimal solution is one of the main challenges in this step.

Probabilistic models are one of the best known models for capturing data generating process and quantifying uncertainties in data using random variables and probability distributions. They are powerful models that are shown to be adaptive and robust and can scale well to large

datasets. However, most probabilistic models have a complex structure. Training them could become challenging commonly due to the presence of intractable integrals in the calculation. To remedy this, they require approximate inference strategies that often results in non-convex optimization problems. The optimization part ensures that the model is the best representative of data or data generating process. The non-convexity of an optimization problem take away the general guarantee on finding a global optimal solution. It will be shown later in this dissertation that inference for a significant number of probabilistic models require solving a non-convex optimization problem.

One of the well-known methods for approximate inference in probabilistic modeling is variational inference. In the Bayesian setting, the target is to learn the true posterior distribution for model parameters given the observations and prior distributions. The main challenge involves marginalization of all the other variables in the model except for the variable of interest. This high-dimensional integral is generally computationally hard, and for many models there is no known polynomial time algorithm for calculating them exactly. Variational inference deals with finding an approximate posterior distribution for Bayesian models where finding the true posterior distribution is analytically or numerically impossible. It assumes a family of distribution for the estimation, and finds the closest member of that family to the true posterior distribution using a distance measure. For many models though, this technique requires solving a non-convex optimization problem that has no general guarantee on reaching a global optimal solution. This dissertation presents a convex relaxation technique for dealing with hardness of the optimization involved in the inference.

The proposed convex relaxation technique is based on semidefinite optimization that has a general applicability to polynomial optimization problem. I will present theoretical foundations and in-depth details of this relaxation in this work. Linear dynamical systems represent the functionality of many real-world physical systems. They can describe the dynamics of a linear time-varying observation which is controlled by a controller unit with quadratic cost function objectives. Designing distributed and decentralized controllers is the goal of many of these systems, which computationally, results in a non-convex optimization problem. In this dissertation, I will further investigate the issues arising in this area and develop a convex relaxation framework to deal with the optimization challenges.

Setting the correct number of model parameters is an important aspect for a good probabilistic model. If there are only a few parameters, model may lack capturing all the essential relations and components in the observations while too many parameters may cause significant complications in learning or overfit to the observations. Non-parametric models are suitable techniques to deal with this issue. They allow the model to learn the appropriate number of parameters to describe the data and make predictions. In this dissertation, I will present my work on designing Bayesian non-parametric models as powerful tools for learning representations of data. Moreover, I will describe the algorithm that we derived to efficiently train the model on the observations and learn the number of model parameters.

Later in this dissertation, I will present my works on designing probabilistic models in combination with deep learning methods for representing sequential data. Sequential datasets comprise a significant portion of resources in the area of machine learning research. Designing models to capture dependencies in sequential datasets are of great interest and have a wide variety of applications in engineering, medicine and statistics. Recent advances in deep learning research has shown exceptional promises in this area. However, they lack interpretability in their general form. To remedy this, I will present my work on mixing probabilistic models with neural network models that results in better performance and expressiveness of the results.

Table of Contents

List of Figures

List of Tables

Acknowledgments

First and foremost, I would like to thank my Ph.D. advisor, Professor John Paisley. I have been extremely lucky and fortunate to have him as my academic advisor in the past few years. He inspired me with his unique vision, encouraged me with his genuine support, and enlighten me with his wisdom and knowledge. I would like to extend my thanks to Professor John Wright, Professor David Blei, Professor Javad Ghaderi, and Professor Daniel Hsu for being part of my thesis committee and providing invaluable advice. Moreover, I have had the greatest opportunity to be a part of Columbia community that have nurtured me and have become my second home.

Ph.D. years have been the toughest years of my life, and none of it could have happened without support and love from my friends and family. I would like to express my deepest gratitudes to my parents, Zhila Mahdian and Saeed Fazelnia. They have given me confidence to believe in my capabilities, guidance to pursue my dreams, and support to wholeheartedly believe that women can achieve anything that men can. They have sacrificed greatly for creating better lives for our family, and I will be forever grateful for that. I would like to thank my siblings, Mohamad and Dorsa, and my uncle Behzad, for always loving me, supporting me, taking care of me, and encouraging me to go distance. Last but absolutely not least, I would like to offer my sincerest thanks to my husband, Saleh. He has been a true loving and supporting partner that anyone could ever wish for. I have been extremely fortunate for having him by my side to believe in me through my failures and disappointments, to give me courage to always do the right thing, and to truly empower me to succeed. His encouragement and vision have been great companions in my Ph.D.. I truly love him, and will always be thankful for his unconditional kindness and immerse support.

To my parents Zhila and Saeed,

to Mohamad, Dorsa, and Behzad,

and to the love of my life Saleh.

Chapter 1

Introduction

Probabilistic models are one of the most powerful techniques for discovering and describing underlying structures in a dataset [48]. They can infer possible models to explain observed data. This approach is flexible to adjust to different variations in data and is robust to tolerate noise in data. Uncertainty in data is a crucial part for this type of models. Due to their probabilistic nature, they are capable of modeling uncertainty and incorporating it for predicting. Generative models assume a generating process from which the data is drawn. Using them enables us to not only train the model and make prediction on unobserved data but also to quantify our uncertainty about the model assumptions and prediction results. Latent variable models are a subcategory of probabilistic models which assume that data is affected by unobserved factors or latent variables. Latent variables could represent the unobserved structures in the data. Latent variable models include factor analysis models, mixture models, and latent class models among others.

Factor analysis models assume that data can be described via a linear combination of unobservable latent variables called factors. Collection of these factors is called dictionary. Sparse coding seeks to decompose data points into combination of a small subset of patterns selected from the dictionary. The goal of dictionary learning is to simultaneously learn these patterns in the dictionary (factors) and the sparse representation of the signals [2]. In this dissertation, I will present my work on dictionary learning and sparse signal representation with Bayesian nonparametric priors in which the number of factors is learned by the model [41].

I derive *probabilistic orthogonal matching pursuit* (PrOMP), a stochastic Expectation Maximization (EM) algorithm for learning sparse data representations, and extend the algorithm to the sparse Bayesian nonparametric dictionary learning task using the beta process. The core EM algorithm provides a new way for doing inference in nonparametric dictionary learning models. Our theoretical analysis builds upon the previous theory for Orthogonal Matching Pursuit (OMP) [31, 102, 128]. Like OMP, PrOMP is a greedy algorithm for regression that iteratively selects columns of a matrix according to a score. This score is based on an atypical use of the EM algorithm and thus optimizes a marginal probability distribution [42].

Mixed membership models provide a probabilistic approach to modeling groups of data through a combination of shared and group-specific parameters [3]. A learning algorithm aims to learn the shared components alongside membership assignment weights for representing each data point using the components. In this dissertation, I will present my work on using mixed membership models when learning sequential data. Deep Neural Networks (DNN) have been widely used for modeling sequential data and have shown great performance in capturing recurring patterns in data. Models of sequential data such as the recurrent neural network (RNN) often implicitly treat a sequence as having a fixed time interval between observations and do not account for group-level effects when multiple sequences are observed. I propose a model for grouped sequential data based on the RNN that accounts for varying time intervals between observations in a sequence by learning a group-level parameter to which each sequence reverts as more time passes between observations. Our approach is motivated by the mixed membership framework, and can be used for dynamic topic modeling-type problems in which the distribution on topics (not the topics themselves) are evolving in time [45].

Probabilistic models with Bayesian hierarchical priors are powerful techniques for creating interpretable and expressive models. However, a major challenge in Bayesian modeling is posterior inference. For many models this requires calculating normalizing integrals that neither have a closed form, nor are solvable numerically in polynomial time. There are two fundamental approaches to addressing the posterior inference problem. One uses Markov chain Monte Carlo (MCMC) sampling techniques that are asymptotically exact. However, these methods tend to be slow compared with point-estimates and not scalable to large datasets [46, 58]. Mean-field variational inference is another approach that approximates the posterior distribution by first defining a simpler family of distributions and then finding a member that is closest to the desired posterior [67] according to the KullbackLeibler (KL) divergence. This turns the inference problem into an optimization problem. This approach introduces new challenges since it mostly results in non-convex optimization problems. In this dissertation, I will present a method to deal with the non-convexities in variational inference (VI) optimization for conjugate prior models that can achieve near globally optimal solutions [43].

The proposed optimization technique is based on convex relaxation and semidefinite programming (SDP). In our approach, an SDP relaxation converts a non-convex polynomial optimization of vector parameters to a convex optimization with matrix parameters via a lifting technique. The exactness of the relaxation can then be interpreted as the existence of a low-rank solution to this SDP. In the last part of this dissertation, I will present theoretical details on this method as well as its implications in linear dynamical systems with decentralized controls [39].

The rest of this dissertation is organized as follows. Chapter 2 presents our results for developing inference algorithm in nonparametric Bayesian modeling. It will be followed by Chapter 3 in which I demonstrate our work on Bayesian nonparametric factor analysis model for dictionary learning and sparse representation. In chapter 4 I introduce our work on probabilistic analysis for sequential data modeling using deep learning techniques. Chapter 5 presents my work on convex relaxation process for variational inference which is followed by Chapter 6 that gives more detail about the relaxation technique and its application for linear time varying datasets. I will conclude this dissertation in the Chapter 7 and bring a discussion for the future directions.

Chapter 2

Probabilistic Orthogonal Matching Pursuit

In this chapter, I present *probabilistic orthogonal matching pursuit* (PrOMP) for learning sparse data representations, and extend the algorithm to the sparse Bayesian nonparametric dictionary learning task using the beta process [42]. Like OMP, PrOMP is a greedy algorithm for regression that iteratively selects columns of a matrix according to a score. This score is based on an atypical use of the EM algorithm and thus optimizes a marginal probability distribution. Our theoretical analysis builds on the previous theory for OMP. We also present experimental results on dictionary learning for the image denoising and compressed sensing magnetic resonance imaging (CS-MRI) problems, where we empirically demonstrate that PrOMP, when used in a dictionary learning algorithm, can lead to improved performance over other sparse dictionary learning and inference approaches by converging to a better local optimal solution. I will expand the applications of PrOMP to more Bayesian nonparametric models in the next chapter.

2.1 Introduction

Sparse data representation is a fundamental problem of signal processing that aims to decompose data into combinations of small subsets of patterns selected from a larger dictionary. Due to its significant applications in areas such as applied mathematics, statistics, and electrical engineering, this field has received much attention. Various methods for sparse data representation and recovery have been proposed using sparse penalties. For instance, least squares with l_1 norm penalties (lasso) [23, 35, 124] are very effective in recovering sparse representation. Many fundamental approaches are build on this l_1 approach [57].

Another important family are matching pursuit (MP) methods, based on iterative greedy algorithms that search for a sparse representation of data [88]. The main idea of MP is to iteratively select columns to incorporate in the representation from a set of given patterns based on correlation with the current residual, after which the residual is updated. This greedy algorithm continues to run until reaching some convergence threshold. For example, orthogonal matching pursuit (OMP) expedites convergence to the final solution by finding the least squares solution on a growing subset of basis functions [102, 128].

The computational simplicity of OMP in addition to its effectiveness for approximating l_0 minimization problems have made it a very appealing method for sparse data representation, and inspired many modifications. For example, stagewise orthogonal matching pursuit (StOMP) [31], allows for multiple simultaneous patterns to be included at once. Issues with threshold selection have been addressed in regularized OMP [92] and compressive sampling OMP [91] that are more robust to noisy conditions and are more efficient in resource usage.

In this chapter, I propose probabilistic orthogonal matching pursuit (PrOMP), which builds upon OMP via a probabilistic model. PrOMP follows an identical algorithmic outline as OMP, but with modified scores calculation and weight (posterior) updates to account for the probabilistic approach. We rigorously derive PrOMP using the expectation-maximization (EM) algorithm, meaning our objective is a marginal likelihood on the sparsity pattern. Our approach to EM, while correct, is not typical since the hidden variables used are growing based on the current active set and thus each E-M step does not correspond to the same breakdown of the marginal. We provide a theoretical analysis of PrOMP inspired by [128] for OMP, with the necessary modification.

I then extend this basic model to the dictionary learning problem using a Bayesian non-parametric beta process prior. In this problem, the additional goals are to learn the patterns themselves based on multiple signals, as well as the number of patterns to use [70, 71, 95]. This approach has proven effective in tasks such as image denoising, inpainting [140], and compressed sensing [64]. Inference for the beta-Bernoulli process has focused on variational methods [95] and Markov Chain Monte Carlo (MCMC) sampling [64, 140] without considering scalability. We address this issue by proposing a novel scalable EM-based algorithm. [2] deals with large data sets by incorporating OMP step used by KSVD, however, our scalable extension is an EM special case of stochastic variational inference [61].

In Section 2.2 I present the problem setup and PrOMP algorithm along with detailed theoretical analysis, making connections to OMP along the way. PrOMP is extended to the sparse dictionary learning problem in Section 2.3 We present the experiments on dictionary learning in Section 5.4.

2.2 Probabilistic Orthogonal Matching Pursuit

I focus on PrOMP for sparse coding of one signal and extend to multiple observations when discussing dictionary learning in Section 2.3. Given a signal $\mathbf{x} \in \mathbb{R}^d$ and matrix $\mathbf{W} \in \mathbb{R}^{d\times K}$, our goal is to find a sparse representation $\mathbf{s}$ for $\mathbf{x}$ such that $\mathbf{x} \approx W\mathbf{s}$. We do this via the model

$$\mathbf{x} \sim \mathcal{N}(W\mathbf{s}, \sigma^2 I). \tag{2.1}$$

Further, we formulate $\mathbf{s} = \mathbf{z} \odot \mathbf{c}$ where $\mathbf{z}$ is a sparse binary coding that indicates the columns of W used in the representation, and $\mathbf{c}$ represents corresponding real-valued weights. Their respective probability distributions are

$$\mathbf{z}_k \sim \text{Bern}(\pi_k), \qquad \mathbf{c} \sim \mathcal{N}(0, \lambda^{-1} I), \tag{2.2}$$

thus $\mathbf{z} \in \{0,1\}^K$ and $\mathbf{c} \in \mathbb{R}^K$. The Bernoulli prior regularization will be a more significant factor for dictionary learning, where each π_k will have a beta prior.

2.2.1 PrOMP via the EM algorithm

In inference procedure, we learn a point estimate for the binary $\mathbf{z}$, and conditional posterior q distribution on $\mathbf{c}$. Following the general regression setup, we assume W is known here and make the appropriate modifications for dictionary learning. In the EM algorithm, the goal is to maximize $p(\mathbf{x}, \mathbf{z})$ over the sparse coding vectors $\mathbf{z}$ with $\mathbf{c}$ treated as the marginalized variables. To this end, we can introduce an arbitrary distribution q on these hidden variables and set up the EM objective function on the log marginal distribution,

$$\ln p(\mathbf{x}, \mathbf{z}) = \underbrace{\mathbb{E}_q\Big[\ln \frac{p(\mathbf{x}, \mathbf{z}, \mathbf{c})}{q(\mathbf{c})}\Big]}_{\mathcal{L}(\mathbf{z})} \;+\; \underbrace{\mathbb{E}_q\Big[\ln \frac{q(\mathbf{c})}{p(\mathbf{c}|\mathbf{x}, \mathbf{z})}\Big]}_{\mathrm{KL}(q\|p)}. \tag{2.3}$$

We note that the sparse coding step based on Eq. (2.3) is not as straightforward as it might appear, since if $z_k = 0$, then the corresponding updated q distribution on c_k will revert to the zero-mean prior, which makes it effectively impossible to set z_k in the following iteration. This is a common problem encountered by Bayesian nonparametric dictionary learning algorithms [115].

However, our less conventional route will be to construct a sequence of the RHS each equal to the LHS. Define the index set $\mathcal{A} \subset \{1, ..., K\}$ and let $\mathbf{c}_{\mathcal{A}}$ denote the subvector of $\mathbf{c}$ defined by $\mathcal{A}$. Then equivalently,

$$\ln p(\mathbf{x}, \mathbf{z}) = \underbrace{\mathbb{E}_q\Big[\ln \frac{p(\mathbf{x}, \mathbf{z}, \mathbf{c}_{\mathcal{A}})}{q(\mathbf{c}_{\mathcal{A}})}\Big]}_{\mathcal{L}_{\mathcal{A}}(\mathbf{z})} + \underbrace{\mathbb{E}_q\Big[\ln \frac{q(\mathbf{c}_{\mathcal{A}})}{p(\mathbf{c}_{\mathcal{A}}|\mathbf{x}, \mathbf{z})}\Big]}_{\mathrm{KL}(q\|p_{\mathcal{A}})}. \tag{2.4}$$

We note that the RHS of Eq. (2.3) & (2.4) are equal, yet $\mathcal{L} \neq \mathcal{L}_{\mathcal{A}}$ and $\mathrm{KL}(q\|p) \neq \mathrm{KL}(q\|p_{\mathcal{A}})$ because they correspond to different joint distributions.[1] Our key observation is that, by setting $q = p_{\mathcal{A}}$ and updating $\mathbf{z}$ according to $\mathcal{L}_{\mathcal{A}}$ in one iteration of EM, we are monotonically increasing $\ln p(\mathbf{x}, \mathbf{z})$ *regardless of the index set used for that particular iteration.* This simply follows from the original proof of EM. We will see here that defining $\mathcal{A}$ and updating $\mathcal{L}_{\mathcal{A}}$ in a particular way will lead to an OMP-like algorithm.

For example, let $\mathcal{A}$ correspond to dimensions in $\mathbf{z}$ set to 1. What the equality in Eq. (2.4) shows is that we can arbitrarily integrate out portions of the vector $\mathbf{c}$ corresponding to those dimensions in which $z_k = 0$ (among other possible options). Furthermore, the conditional posterior $p(\mathbf{c}_{\mathcal{A}}|\mathbf{x}, \mathbf{z})$ is easily shown to be Gaussian and the calculation of $\mathcal{L}_{\mathcal{A}}$ is straightforward. We use this observation to derive a two step greedy algorithm for jointly learning $\mathbf{z}$ and $q(\mathbf{c})$.

[1] $p(\mathbf{x}, \mathbf{z}, \mathbf{c}_{\mathcal{A}})$ is a marginal of $p(\mathbf{x}, \mathbf{z}, \mathbf{c})$ over a subset of $\mathbf{c}$.

Our two step greedy procedure first calculates the E-step in Eq. (2.4) using the current $\mathcal{A}$. It then picks the dimension of $\mathbf{z}$ to set to 1 that increases $\mathcal{L}_\mathcal{A}$ the most. Calling this dimension j, it finally increments $\mathcal{A} \leftarrow \mathcal{A} \cup \{j\}$. The next iteration then recomputes $\mathcal{L}_\mathcal{A}$ *based on the augmented set* $\mathcal{A}$. Therefore, each iteration optimizes over a different function $\mathcal{L}_\mathcal{A}$ using a $q(\mathbf{c}_\mathcal{A})$ that is growing in dimensionality. If including no dimension of $\mathbf{z}$ can increase $\mathcal{L}_\mathcal{A}$, then the algorithm terminates. Unlike OMP, for which this never can happen and thus termination criteria are necessary, the prior regularization of our Bayesian model allows for automatic termination based on the marginal likelihood.

2.2.2 The PrOMP algorithm

As discussed previously, regardless of the index set $\mathcal{A}$, by setting $q(\mathbf{c}_\mathcal{A}) = p(\mathbf{c}_\mathcal{A}|\mathbf{x}, \mathbf{z})$ and updating $\mathbf{z}$ such that $\mathcal{L}_\mathcal{A}(\mathbf{z})$ increases, we are guaranteed to monotonically improve $\ln p(\mathbf{x}, \mathbf{z})$. Our proposal is to greedily add columns of $\mathbf{W}$ by setting the corresponding dimensions of $\mathbf{z}$ to 1 based on how much $\mathcal{L}_\mathcal{A}$ improves. The score for dimension j is denoted by $\xi_j^+ - \xi_j^-$. This is the difference between $\mathcal{L}_\mathcal{A}$ for $\mathbf{z}_j = 1$ (+) or 0 (-). For all j, ξ_j^- will be the same for the current iteration, but is included as a stopping criterion since it indicates whether there are no j that increases $\mathcal{L}_\mathcal{A}$, at which point the algorithm terminates. The PrOMP outline is shown in Algorithm 1 with the equations below.

For the E-step, we have a Gaussian conditional posterior on the subset of $\mathbf{c}$ given by the active dimensions in $\mathbf{z}$, $q(\mathbf{c}_\mathcal{A}) = \mathcal{N}(\mathbf{c}_\mathcal{A}|\mu_\mathcal{A}, \Sigma_\mathcal{A})$ where

$$\Sigma_\mathcal{A} = (\lambda I + \mathbf{W}_\mathcal{A}^\top \mathbf{W}_\mathcal{A}/\sigma^2)^{-1},\ \mu_\mathcal{A} = \Sigma_\mathcal{A} \mathbf{W}_\mathcal{A}^\top \mathbf{x}/\sigma^2. \tag{2.5}$$

Here we let $\mathbf{W}_\mathcal{A}$ denote the submatrix of $\mathbf{W}$ formed by selecting the columns of $\mathbf{W}$ indexed by $\mathcal{A}$. For a particular iteration, the likelihood $p(\mathbf{x}|\mathbf{z}, \mathbf{c}_\mathcal{A})$ with hidden data $\mathbf{c}_\mathcal{A}$ is also Gaussian where the appropriate dimensions in $\mathbf{c}$ have been integrated out of the joint likelihood term. Since those dimension correspond to $\mathbf{z}_j = 0$ (as defined by the set $\mathcal{A}$), the result gives

$$\begin{aligned} p(\mathbf{x}|\mathbf{c}_\mathcal{A}, \mathbf{z}_j = 1) &= \mathcal{N}(x|\mathbf{W}_\mathcal{A} c_\mathcal{A}, \sigma^2 I + \lambda^{-1} \mathbf{w}_j \mathbf{w}_j^\top), \\ p(\mathbf{x}|\mathbf{c}_\mathcal{A}, \mathbf{z}_j = 0) &= \mathcal{N}(x|\mathbf{W}_\mathcal{A} c_\mathcal{A}, \sigma^2 I). \end{aligned} \tag{2.6}$$

Algorithm 1 PrOMP

```
1: input: $\mathbf{W}$ and values (later, distributions) for each $\pi_k$
2: output: Coding set $\mathcal{A}$ and weight distribution $q(\mathbf{c}_\mathcal{A})$
3: Set $\mathcal{A} = \emptyset$
4: For each $j$, set $\xi_j^{\pm} = \ln p(\mathbf{x}, \mathbf{z}_j = \{0,1\})$    (Eq. 2.7)
5: while $\max_j \ \xi_j^+ - \xi_j^- \ > \ 0$ do
6:     Set $j' = \arg\max_j \ \xi_j^+ - \xi_j^-$
7:     Set $\mathcal{A} \leftarrow \mathcal{A} \cup \{j'\}$, $\mathbf{z}_{j'} = 1$ and $\xi_{j'}^+ = -\infty$
8:     Set $q(\mathbf{c}_\mathcal{A}) = p(\mathbf{c}_\mathcal{A}|\mathbf{x}, \mathbf{z})$    (Eq. 2.5)
9:     For each $j \notin \mathcal{A}$, update    (Eq. 2.7)
           $\xi_j^+ = \mathbb{E}_q[\ln p(\mathbf{x}|\mathbf{c}_\mathcal{A}, \mathbf{z}_j = 1)] + \ln \pi_j,$
           $\xi_j^- = \mathbb{E}_q[\ln p(\mathbf{x}|\mathbf{c}_\mathcal{A}, \mathbf{z}_j = 0)] + \ln(1 - \pi_j)$
10: end while
```

Thus, for a given active set $\mathcal{A}$, the score for activating a particular dimension j of $\mathbf{z}$ can be found. Define the approximation residual to be $r_\mathcal{A} = \mathbf{x} - \mathbf{W}_\mathcal{A}\mu_\mathcal{A}$. Then

$$\begin{aligned} \xi_j^+ - \xi_j^- &= \mathbb{E}_q\left[\ln \frac{p(\mathbf{x}, \mathbf{z}_j = 1|\mathbf{c}_\mathcal{A})}{p(\mathbf{x}, \mathbf{z}_j = 0|\mathbf{c}_\mathcal{A})}\right] \\ &= \frac{1}{2\sigma^2}\frac{(r_\mathcal{A}^\top \mathbf{w}_j)^2 + \mathbf{w}_j^\top \mathbf{W}_\mathcal{A}\Sigma_\mathcal{A}\mathbf{W}_\mathcal{A}^\top \mathbf{w}_j}{\lambda\sigma^2 + \mathbf{w}_j^\top \mathbf{w}_j} \\ &\quad - \ln(1 + \frac{\mathbf{w}_j^\top \mathbf{w}_j}{\lambda\sigma^2}) + \ln \frac{\pi_j}{1 - \pi_j}. \end{aligned} \tag{2.7}$$

The difference between PrOMP and OMP consists of the following modification: If $\mu_\mathcal{A} \equiv (\mathbf{W}_\mathcal{A}^\top \mathbf{W}_\mathcal{A})^{-1}\mathbf{W}_\mathcal{A}^\top \mathbf{x}$ and $\xi_j^+ - \xi_j^- \equiv (r_\mathcal{A}^\top \mathbf{w}_j)^2/(\mathbf{w}_j^\top \mathbf{w}_j)$, then the OMP algorithm results. The additional values are probabilistic terms accounting for uncertainty in $\mathbf{c}_\mathcal{A}$, as well as the penalty term π_j for dimension j. As mentioned above, in our extension to dictionary learning we will see how inferring π_j with a sparse beta process prior can make this a nonparametric algorithm.

2.2.3 Theoretical guarantees for PrOMP

As seen, PrOMP consists of a modification to OMP that accounts for probabilistic uncertainty. Like the extensive theory developed for OMP, we modify these results for PrOMP by considering two general problems: (1) The sparsest representation problem, whose goal is to identify the representation of the input signal that uses the least number of atoms; (2) m-sparse problem in which the signal x is required to have an m-term representation and the goal is to identify m terms in the dictionary to represent $\mathbf{x}$.

Theorem 2.1 (Sparsest Representation). *Assume that the sparsest representation is unique. Then, PrOMP will find that representation if*

$$\max_{j \notin \mathcal{A}_{opt}} d \|\mathbf{W}^{+}_{\mathcal{A}_{opt}} \mathbf{w}_j\|_2 < 1, \tag{2.8}$$

and provided that $\ln \frac{\pi_j}{1-\pi_j} \geq \ln \frac{\pi'_j}{1-\pi'_j}$ *and* $\|\mathbf{w}_j\|_2 \leq \|\mathbf{w}_{j'}\|_2$ *for all* j *and* j' *in the optimal set and outside the optimal set, respectively.*

We note that the condition in Eq. 2.8 has resemblance to the Exact Recovery Condition (ERC) in [128].

Proof 2.1. *Let us define* $W_{\overline{\mathcal{A}}_{opt}} \triangleq W_{j \notin \mathcal{A}_{opt}}$. *According to Algorithm 1, at step k, in order to select an index, we need to find a j for which* $\xi_j^+ - \xi_j^-$ *is positive and has the highest value among all of the candidates. Suppose that we have performed k steps of PrOMP and achieved* $\mathcal{A}_k$. *Define*

$$\rho(r_{\mathcal{A}_k}) \triangleq \frac{\max_{j' \in \overline{\mathcal{A}}_{opt}} (\xi_{j'}^+ - \xi_{j'}^-)}{\max_{j \in \mathcal{A}_{opt}} (\xi_j^+ - \xi_j^-)} = \tag{2.9}$$

$$\frac{\max_{j' \in \overline{\mathcal{A}}_{opt}} \mathbf{w}_{j'}^\top (r_{\mathcal{A}_k} r_{\mathcal{A}_k}^\top + \mathbf{W}_{\mathcal{A}_k} \Sigma_{\mathcal{A}_k} \mathbf{W}_{\mathcal{A}_k}^\top)^\top \mathbf{w}_{j'} + f(j')}{\max_{j \in \mathcal{A}_{opt}} \mathbf{w}_j^\top (r_{\mathcal{A}_k} r_{\mathcal{A}_k}^\top + \mathbf{W}_{\mathcal{A}_k} \Sigma_{\mathcal{A}_k} \mathbf{W}_{\mathcal{A}_k}^\top)^\top \mathbf{w}_j + f(j)}.$$

Note that $\max_{j \in \mathcal{A}_{opt}} (\xi_j^+ - \xi_j^-) = \max_{j \in \mathcal{A}_{opt} \setminus \mathcal{A}_k} (\xi_j^+ - \xi_j^-)$ *as we set* $\xi_j^+ = -\infty$ *if j have been previously chosen and the search is only over positive score values inside or outside the optimal set. Also, the conditions of this theorem guarantee that* $f(j') < f(j)$. *PrOMP will pick an index from the optimal set at the next step if and only if* $\rho(r_{\mathcal{A}_k}) < 1$. *PrOMP will not pick the same* *index twice, and j and j' are not previously chosen. So, in order to have $\rho(r_{\mathcal{A}_k}) < 1$, it suffices to have*

$$\frac{\max_{j' \in \overline{\mathcal{A}}_{opt}} \mathbf{w}_{j'}^{\top} (r_{\mathcal{A}_k} r_{\mathcal{A}_k}^{\top} + \mathbf{W}_{\mathcal{A}_k} \Sigma_{\mathcal{A}_k} \mathbf{W}_{\mathcal{A}_k}^{\top})^{\top} \mathbf{w}_{j'}}{\max_{j \in \mathcal{A}_{opt}} \mathbf{w}_{j}^{\top} (r_{\mathcal{A}_k} r_{\mathcal{A}_k}^{\top} + \mathbf{W}_{\mathcal{A}_k} \Sigma_{\mathcal{A}_k} \mathbf{W}_{\mathcal{A}_k}^{\top})^{\top} \mathbf{w}_{j}} < 1. \tag{2.10}$$

Let us define $r_{\mathcal{A}_k} r_{\mathcal{A}_k}^{\top} + \mathbf{W}_{\mathcal{A}_k} \Sigma_{\mathcal{A}_k} \mathbf{W}_{\mathcal{A}_k}^{\top} = l_k^{\top} l_k$. Note that this decomposition is possible due to the positive definiteness of the left hand side. The left hand side of Eq. 2.10 equals the following:

$$\begin{aligned}
\frac{\|l_k \mathbf{W}_{\overline{\mathcal{A}}_{opt}}\|_{2,\infty}}{\|l_k \mathbf{W}_{\mathcal{A}_{opt}}\|_{2,\infty}} &= \frac{\|l_k \mathbf{W}_{\mathcal{A}_{opt}} \mathbf{W}_{\mathcal{A}_{opt}}^{+} \mathbf{W}_{\overline{\mathcal{A}}_{opt}}\|_{2,\infty}}{\|l_k \mathbf{W}_{\mathcal{A}_{opt}}\|_{2,\infty}} \\
&\leq \frac{d\ \|l_k \mathbf{W}_{\mathcal{A}_{opt}}\|_{2,\infty} \|\mathbf{W}_{\mathcal{A}_{opt}}^{+} \mathbf{W}_{\overline{\mathcal{A}}_{opt}}\|_{2,\infty}}{\|l_k \mathbf{W}_{\mathcal{A}_{opt}}\|_{2,\infty}} \\
&= d\ \|\mathbf{W}_{\mathcal{A}_{opt}}^{+} \mathbf{W}_{\overline{\mathcal{A}}_{opt}}\|_{2,\infty} \\
&= \max_{j \notin \mathcal{A}_{opt}} \ d \|\mathbf{W}_{\mathcal{A}_{opt}}^{+} \mathbf{w}_j\|_2
\end{aligned} \tag{2.11}$$

where $\|\cdot\|_{2,\infty}$ is the maximum of the column-wise L2 norm for a matrix. Hence, if $\max_{j \notin \mathcal{A}_{opt}} d \|\mathbf{W}_{\mathcal{A}_{opt}}^{+} \mathbf{w}_j\|_2 < 1$, *this guarantees that PrOMP will pick an index from the optimal set.* □

Theorem 2.2. *[128] Let m indicate the number of columns in$\mathbf{W}_{\mathcal{A}_{opt}}$. Then, condition in Theorem 2.1 holds if*

$$d\omega_1(m-1) + d\omega_1(m) < 1. \tag{2.12}$$

In other words, this condition will recover every signal with m-term representation.

where $\omega_1(m)$ represents the cumulative coherence function defined as $\omega_1(m) \triangleq \max_{\Lambda} \max_{j \in \Omega \setminus \Lambda} \|\mathbf{W}_{\Lambda}^{\top} \mathbf{W}_j\|_1$ with Λ being an index set of m columns of $\mathbf{W}$, and Ω showing all the indices . $\omega_1(0)$ is set to 0. We should point out that the proof of 2.2 follows the same steps as the ones in [128] noting that L2 norm of a vector is upper bounded by its L1 norm value.

2.3 BNP dictionary learning

In this section, we expand the sparse representation problem to learning the matrix $\mathbf{W}$ and probabilities π_k using Bayesian nonparametrics (BNP) similar to previous approaches, with the contribution being the novel inference algorithm. Extending the problem to N signals, BNP dictionary learning uses the generative process

$$\begin{aligned}
&\mathbf{x}_n \sim \mathcal{N}(\mathbf{W}(\mathbf{z}_n \odot \mathbf{c}_n), \sigma^2 I), \\
&\mathbf{c}_n \sim \mathcal{N}(0, \lambda^{-1} I),\ z_{n,k} \sim \text{Bern}(\pi_k), \\
&\mathbf{w}_k \sim \mathcal{N}(0, \eta^{-1} I),\ \pi_k \sim \text{beta}(\alpha\tfrac{\gamma}{K}, \alpha(1-\tfrac{\gamma}{K})),
\end{aligned} \tag{2.13}$$

for $\alpha, \gamma > 0$ and $k = 1, \ldots, K$ in the limit $K \to \infty$.When K is finite but large for practical implementation, this model gives a good approximation in that it will learn a subset of indexes k such that π_k is substantial.

2.3.1 Inference for BNP dictionary learning

In the inference process, we learn point estimates for $\mathbf{W}$ and each $\mathbf{z} = \{\mathbf{z}_n\}$. We propose a new MAP-EM algorithm that uses PrOMP to learn sparse codings of $\mathbf{x} = \{\mathbf{x}_n\}$ with π and $\mathbf{c} = \{\mathbf{c}_n\}$ the hidden variables. Similar to the discussion in Section 2.2, the EM algorithm now aims to maximize $p(\mathbf{x}, \mathbf{z}, \mathbf{W})$ over $\mathbf{z}$ and $\mathbf{W}$ using the equality

$$\begin{aligned}
\ln p(\mathbf{x}, \mathbf{z}, \mathbf{W}) \;=\;& \underbrace{\mathbb{E}_q\Big[\ln \frac{p(\mathbf{x}, \mathbf{z}, \mathbf{W}, \mathbf{c}, \pi)}{q(\mathbf{c}, \pi)}\Big]}_{\mathcal{L}(\mathbf{z},\mathbf{W})} \\
+\;& \underbrace{\mathbb{E}_q\Big[\ln \frac{q(\mathbf{c}, \pi)}{p(\mathbf{c}, \pi|\mathbf{x}, \mathbf{z}, \mathbf{W})}\Big]}_{\text{KL}(q\|p)}
\end{aligned} \tag{2.14}$$

We give an outline in Algorithm 2 and the relevant new equations below. We note that PrOMP requires the modification that $\ln \pi_j$ is replaced with $\mathbb{E}_q[\ln \pi_j]$, and similarly for $\ln(1 - \pi_j)$, since π_j is being marginalized with EM. We discuss these values below.

First we observe that $\mathbf{c}$ and π are conditionally independent given $\mathbf{W}$. Furthermore, the $\mathbf{c}_n$ and π_k are also conditionally independent in their posterior. Thus we have

$$q(\mathbf{c}, \pi) = \Big[\textstyle\prod_{n=1}^N q(\mathbf{c}_n)\Big] \Big[\textstyle\prod_{k=1}^K q(\pi_k)\Big]. \tag{2.15}$$

Since this follows the factorization of $p(\mathbf{c}, \pi|\mathbf{x}, \mathbf{z}, \mathbf{W})$ we are not doing mean field variational inference.

Algorithm 2 Sparse Dictionary Learning with PrOMP

```
1: Initialize dictionary $\mathbf{W}$.
2: while not converged do
3:     Update each $\mathbf{z}_n$ and $q(\mathbf{c}_n)$                (Algorithm 1*)
4:     Update $\mathbf{W}$ and $q(\pi)$ with EM                  (Eqs. 2.16 & 2.17)
5: end while
```

*Replace $\ln \frac{\pi_j}{1-\pi_j}$ with $\mathbb{E}_q\big[\ln \frac{\pi_j}{1-\pi_j}\big]$. See Sec. 2.3.1 for discussion.

After performing PrOMP to update each $\mathbf{z}_n$ and $q(\mathbf{c}_n)$, we update the global variables $\mathbf{W}$ and π using a straightforward application of EM as follows:

E-Step: For each k, update $q(\pi_k) = \text{beta}(a_k, b_k)$, where

$$a_k = \tfrac{\alpha\gamma}{K} + \sum_{n=1}^{N} z_{n,k}, \quad b_k = \alpha(1 - \tfrac{\gamma}{K}) + \sum_{n=1}^{N} (1 - z_{n,k}) \tag{2.16}$$

and using $q(\mathbf{c}_n)$ from PrOMP, calculate $\mathcal{L}(\mathbf{z}, \mathbf{W})$. We note that for PrOMP we now use $\mathbb{E}_q\big[\frac{\pi_k}{1-\pi_k}\big] = \psi(a_k) - \psi(b_k)$, where $\psi(\cdot)$ is the digamma function.

M-Step: Let $\widehat{\mathbf{z}}_n = \text{diag}(\mathbf{z}_n)$. $\arg\max_{\mathbf{W}} \mathcal{L}(\mathbf{z}, \mathbf{W})$ gives

$$\mathbf{W} = \Big[\sum_{n=1}^{N} \mathbf{x}_n \mu_n^\top \widehat{\mathbf{z}}_n\Big]\Big[\eta\sigma^2 I + \sum_{n=1}^{N} \widehat{\mathbf{z}}_n(\mu_n\mu_n^\top + \Sigma_n)\widehat{\mathbf{z}}_n\Big]^{-1}. \tag{2.17}$$

After updating $\mathbf{W}$ and each $q(\pi_k)$, we return to the PrOMP algorithm to sparse code each $\mathbf{x}_n$ by updating $\mathbf{z}_n$ and $q(\mathbf{c}_n)$.

Next, we present some theoretical results when learning dictionary $\mathbf{W}$. We note that this optimizes the function

$$\begin{aligned}\mathcal{L}(\mathbf{z}, \mathbf{W}) = &- \frac{1}{2\sigma^2}\sum_{n=1}^{N} \|\mathbf{x}_n - \mathbf{W}(\mathbf{z}_n \odot \mu_n)\|^2 \\ &- \frac{1}{2\sigma^2}\sum_{n=1}^{N} \text{trace}(\Sigma_n(\mathbf{W}^\top\mathbf{W} \odot \mathbf{z}_n\mathbf{z}_n^\top)) \\ &- \frac{\eta}{2}\text{trace}(\mathbf{W}\mathbf{W}^\top) + \text{const.}\end{aligned} \tag{2.18}$$

The following lemma is useful for our results.

Lemma 2.1. *[108] For a general real-valued matrix M, minimizing trace($MM^\top$) have impact on reducing the rank of M and penalizes its largest singular value.*

Proposition 2.1. *In light of Lemma 2.1 and Theorem 2.1 for PrOMP, the terms in the second and third lines are penalties with the following properties:*

- *The first penalty reduces $\mathcal{C}_k$ and increases $\omega_1(m)$, which results in tightening the approximation bound. It also lowers the rank of $\mathbf{W}diag(\mathbf{z}_n)\sqrt{\Sigma_n}$ resulting in more dependency among columns of this matrix, and more robustness regarding selection of columns when evaluating $\xi_j^+ - \xi_j^-$ in the sparse coding step. In other words, when iteratively performing PrOMP algorithm and dictionary EM update, the chances of changing dictionary elements selected decreases.*
- *The second penalty term reduces the rank of $\mathbf{W}$. This results in more dependency between columns of $\mathbf{W}$ and limits the candidate dictionary elements since PrOMP will not pick a column of $\mathbf{W}$ that is linearly dependent to a column previously selected.*

Table 2.1: SSIM | PSNR performance. K-SVD sees the ground truth when choosing the number of dictionary elements for each image.

		$\sigma = 10$	$\sigma = 15$	$\sigma = 20$	$\sigma = 25$	$\sigma = 50$
House	BPFA	**0.941** | **35.58**	**0.914** | **33.64**	**0.887** | 32.32	**0.872** | 31.20	**0.811** | 27.64
	K-SVD	0.924 | 35.43	0.888 | 33.56	0.879 | **32.61**	0.864 | **31.51**	0.800 | **28.01**
	MFA	0.880 | 29.05	0.868 | 28.97	0.859 | 26.77	0.847 | 26.52	0.803 | 25.82
Lena	BPFA	**0.937** | 34.64	**0.919** | 32.99	**0.901** | **31.70**	**0.885** | **30.66**	**0.806** | **27.37**
	K-SVD	0.932 | **34.87**	0.906 | **33.01**	0.878 | 31.53	0.857 | 30.48	0.761 | 26.82
	MFA	0.865 | 29.34	0.856 | 28.72	0.850 | 27.02	0.839 | 26.99	0.783 | 26.01
Barbara	BPFA	**0.959** | 33.90	**0.943** | **32.04**	**0.926** | **30.52**	**0.907** | **29.27**	**0.800** | **25.48**
	K-SVD	0.956 | **33.96**	0.934 | 31.72	0.909 | 30.16	0.882 | 28.80	0.735 | 24.62
	MFA	0.902 | 29.11	0.894 | 26.14	0.853 | 24.70	0.822 | 24.34	0.767 | 24.11
Boat	BPFA	**0.887** | 33.54	**0.850** | **31.71**	**0.818** | **30.40**	**0.785** | **29.32**	**0.675** | **26.08**
	K-SVD	0.881 | **33.56**	0.840 | 31.70	0.810 | 30.37	0.775 | 29.30	0.659 | **26.08**
	MFA	0.872 | 29.80	0.814 | 27.55	0.793 | 26.67	0.747 | 25.38	0.625 | 24.79

2.4 Experiments

We consider two problems: image denoising and compressed sensing for MRI, which is an application of image denoising. Our experiments show the benefits of PrOMP in the BNP dictionary learning setting compared with K-SVD, the classic dictionary learning algorithm, and also compared with the superior MCMC technique, but applied to an empirically worse inference approach to the same model.

2.4.1 Image Denoising

We compare BPFA using PrOMP with K-SVD [2] and mixtures of factor analyzers (MFA) [49] on an image denoising task. All algorithms performed better than total variation denoising [52], which we omit for space. We observe that BPFA is a Bayesian nonparametric extension of K-SVD that uses OMP instead of PrOMP. MFA is a mixture of non-sparse dictionary learning models (i.e., $\mathbf{z}$ is removed) where each signal chooses one model.

2.4.1.1 Setup

We use four classic test images shown in Figure 2.1. To each image, we add white Gaussian noise with standard deviation $\sigma \in \{10, 15, 20, 25, 50\}$. To quantitatively assess performance, we use the Structural Similarity Index Measure (SSIM) and PSNR [134] of the denoised image to the ground truth. For K-SVD, we use the code provided by [2] and set the dictionary size to give the best results (more discussed later). In all algorithms, we use the technique in [83] to set the noise parameter. To set the parameters for MFA, we follow the suggestions of [49]. Therefore, all the algorithms are compared under the same noise assumption, which we observed was close to the ground truth. For BPFA, K-SVD and MFA, we extracted 16×16 patches from each image using shifts of one pixel, which overall produced the best results for all algorithms compared with 8×8 and 12×12. We ran the algorithms with the following settings: For BPFA $K = 256$ and K-SVD $K = 200$. (Figure 2.2 shows that BPFA used less than 256 elements.). We set $\eta = 255^2$ and for MFA we set the mixture size to $M = 50$ and $D = 20$ to be the subspace size of each mixture.

Figure 2.1: Denoising images: House, Lena, Barbara, Boat.

2.4.1.2 Denoising results

In Table 2.1 we show quantitative results for image denoising. We see that the sparse coding algorithms perform similarly, but overall augmenting with BPFA improved the denoising result over K-SVD. We notice that MFA performs significantly worse than the dictionary learning methods, which shows the advantage of sparse coding versus clustering. However, a key observation is that since K-SVD is not a BNP model, all initialized dictionary elements will be used by the model. Here the number of dictionary elements set for K-SVD changes according to the best result. In practice this would require cross-validation, which is more difficult than our unrealistic approach, which allowed K-SVD to compare with the ground truth when setting this value. We attribute the improvement of BPFA over K-SVD at its best setting to the additional probabilistic structure of the model.

To emphasize this, in Figure 2.2, we show the SSIM results for K-SVD and BPFA for different images and noise settings as a function of dictionary elements set for K-SVD. The results for K-SVD are shown in blue with shaded uncertainty calculated over 20 random initializations. The solid red line shows performance of BPFA. Since BPFA learns the number of dictionary elements we indicate this number with red dashed lines. As it evident, the performance of K-SVD is highly influenced by the number of elements chosen for the dictionary.

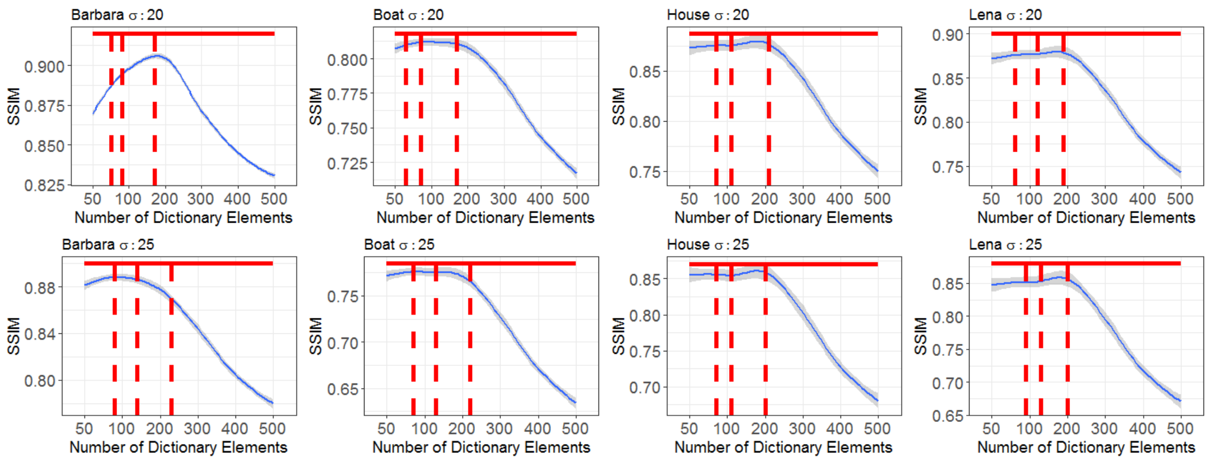


Figure 2.2: (Blue: K-SVD, Red: BPFA) Average SSIM results for different images and noise settings as a function of number of dictionary elements for K-SVD, each repeated for 20 trials. The red line indicates the results for BPFA which is not a function of the x-axis. The dashed red lines are the number of dictionary elements in BPFA responsible for representing 95%, 97% and 99% of all input data.

2.4.2 Compressed Sensing MRI

Magnetic resonance imaging (MRI) is a widely used non-invasive medical imaging method that provides high resolutions images from the anatomy [84]. However, its data acquisition process is slow due to physiological and hardware constraints. The data is produced sequentially in the Fourier measurement domain called k-space, and one way to speed up the process is to undersample from this space. However, this violates the Nyquist theorem, and causes aliasing effects in the reconstructed image when the missing values are replaced with zeros. Compressed sensing has had a major impact on MRI (CS-MRI) and allowed for signal reconstruction from very few samples if the signal is sparse in a particular transform domain [84].

Sparse dictionary learning aims learn this transform domain for CS-MRI directly for the data being considered [24, 107]. Let $\mathbf{x}_n \in \mathbb{C}^d$ be a patch of an MR image X in a vectorized form. Let $\mathcal{F} \in \mathbb{C}^{u \times d}$ be the undersampled Fourier encoding matrix and $\mathbf{y} = \mathcal{F}X \in \mathbb{C}^u$ represent

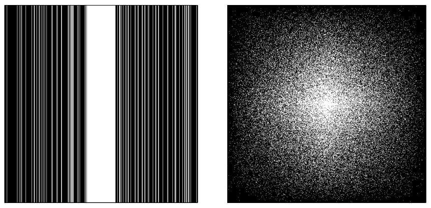


Figure 2.3: Two masks considered. Left: 1D Cartesian sampling with random phase encodes (30% sample rate shown); Right: 2D random sampling (25% sample rate shown).

the sub-sampled set of k-space measurements. The goal of CS-MRI is to estimate X from the small fraction of k-space measurements y. The dictionary learning approach to this optimization problem is to find a dictionary $\mathbf{W}$ as well as sparse representation $\mathbf{s}_n$ for each $\mathbf{x}_n$ such that $\mathbf{x}_n \approxeq \mathbf{W}\mathbf{s}_n$ and $\mathcal{F}\widehat{X} \approx \mathbf{y}$, with $\widehat{X}$ the dictionary learning reconstruction of X.

[64] consider BNP dictionary learning for this task using BPFA, and show that this outperforms a similar method based on K-SVD [107]. That paper uses MCMC sampling for dictionary learning, which technically should give better results than EM for the same model. However, a major drawback of MCMC, as well as the variational inference approach of [95], is that updating each dimension of $\mathbf{z}_n$ is conditioned on the current values of the other dimensions. This can lead to slow mixing, i.e., bad local optimal solutions, because selecting a dictionary element depends on which other elements are currently selected. PrOMP for BPFA (and OMP for K-SVD) are fundamentally different in their sparse coding in that all elements are initialized to zero and greedily selected. We compare this PrOMP version of BPFA with the MCMC sampler used by [64] to illustrate that this greedy EM approach to dictionary learning improves the MCMC approach, which is state-of-the-art for this BNP model.

We experiment on two publicly available 512×512 MRI of a shoulder and lumbar. We apply the masks in Figure 2.3 to subsample MRI in the Fourier domain. We consider various sampling rates for each mask and different sample noise settings. Figure 2.4 shows one example of the reconstruction replacing the missing values with zeros. For all images, we extract 6×6 patches and set other parameters according to [64].

It is well-known that CS inversion is closely related to image denoising, with the noise due to image artifacts from subsampling; [64] provide a discussion on this in the context of dictionary

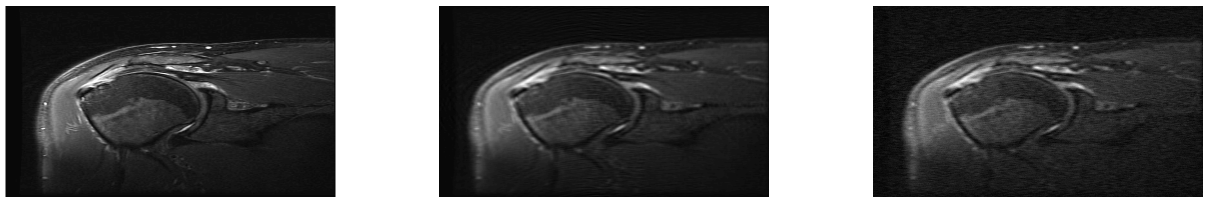


Figure 2.4: (left) Original Shoulder, (middle) Shoulder distorted by Cartesian 35% mask, (right) Shoulder distorted by Cartesian 35% mask and noise with $\sigma = 0.05$.

learning for CS-MRI. Therefore, comparing PrOMP-EM with MCMC for CS-MRI is essentially a comparison of how the denoising abilities of these two inference approaches to the same model translates to a particular task. We first consider the noise-free setting. Figure 2.5 compares the results of PrOMP and MCMC Gibbs sampling for different MRI, masks and sampling percentages. It is evident that PrOMP performs better at reconstructing original image. We also evaluate performance on the same setup, but with additive sampling noise with standard deviation $\sigma = 0.05$. (The original MRI was scaled to $[0, 1]$. Other settings of σ showed the same pattern.) For these experiments we use the original noise-free MRI as ground truth. Figure 2.6 shows the PSNR results for lumbar image for different masks, sampling percentages and noise values. It is evident that PrOMP performs better in reconstructing the original image as well as denoising it. Figure 2.7 shows that PrOMP is capable of learning a sparser representation than MCMC with far fewer dictionary elements required.

2.5 Conclusion

I proposed probabilistic orthogonal matching pursuit (PrOMP) for sparse data representation. Our probabilistic approach extends orthogonal matching pursuit (OMP), making it suitable for statistical dictionary learning models with Bayesian nonparametric priors. We derived theory for PrOMP similar to that of OMP, and discussed how PrOMP can improve existing dictionary learning models. We evaluated the performance on image denoising and compresses sensing for

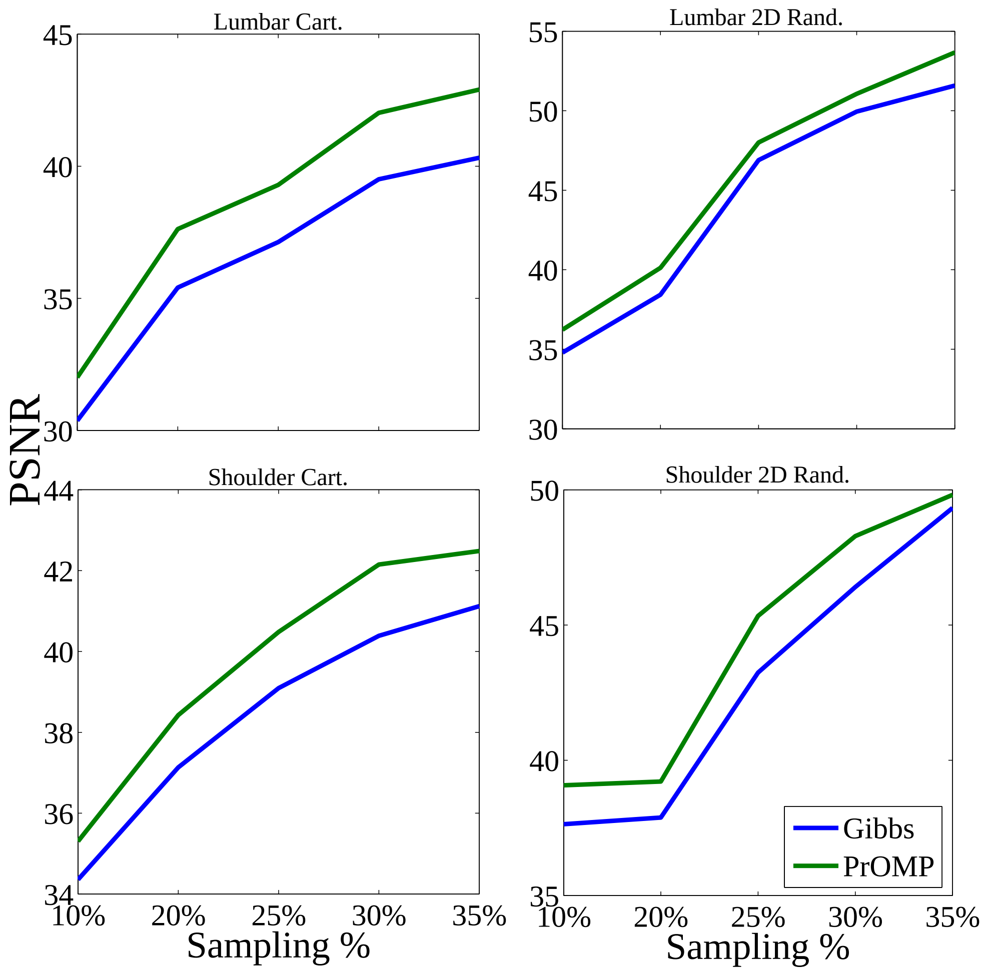


Figure 2.5: PSNR results in the noiseless setting for different MRI, masks and sampling percentages. PrOMP for EM outperforms MCMC for the same BNP dictionary learning model.

magnetic resonance imaging (CS-MRI), showing that PrOMP for BPFA improves the classic K-SVD model, as well as MCMC sampling for the same BNP dictionary learning model.

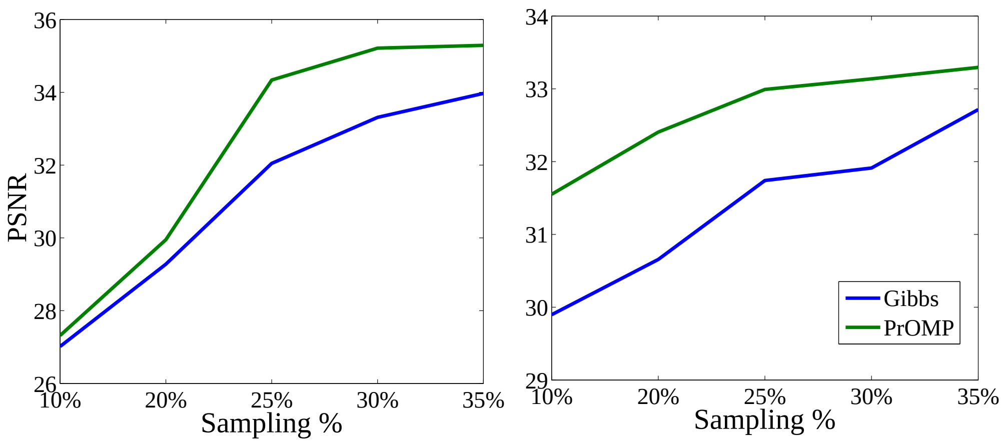


Figure 2.6: PSNR results with additive sampling noise ($\sigma = 0.05$) for the lumbar MRI, using the Cartesian (left) and Random (right) masks and different sampling percentages. PrOMP outperforms MCMC for the same BNP dictionary learning model.

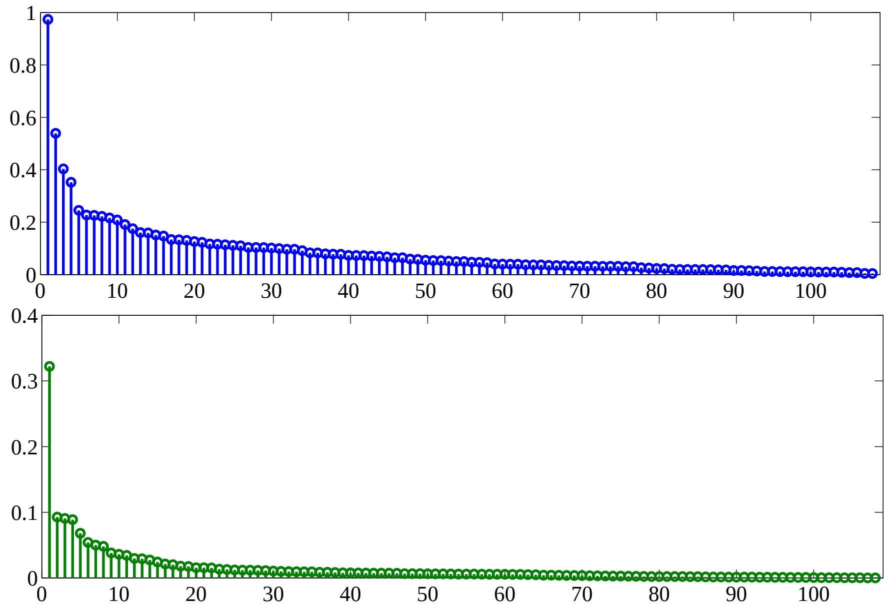


Figure 2.7: Stem plots of π_k (sorted) for lumbar MRI with Cartesian 25% sampling mask. PrOMP (bottom) shows the expectation and MCMC (top) shows the best performing iteration. PrOMP learns a much sparser representation for this model.

Chapter 3

Beta Process Subspace Analysis

In this chapter, I present a new model for latent subspace analysis in which the number of subspaces and dimensionality of each subspace are inferred using Bayesian nonparametric priors [41]. Latent membership models enable us to discover underlying structures in a dataset where in this chapter the latent members are subspaces.In our formulation, a beta process prior allows for an unbounded number of subspaces, while gamma process priors on the variances of dictionary elements in each subspace allow for unbounded subspace dimensionality. We call our model *beta process subspace analysis* (BPSA), which can be thought of as a subspace extension of a related factor analysis model that uses the beta process. We derive a scalable EM algorithm and demonstrate performance on image denoising tasks and learning on large image dataset.

3.1 Introduction

Latent membership models are useful techniques in discovering and describing underlying structure in a dataset. Latent members of the model are designed such that each observation can be constructed from those members. In this work we focus on learning these latent members in the format of a dictionary that consists of subspaces, as well as sparse coding for the observations using dictionary subspaces.

Sparse coding seeks to decompose a signal into a combination of a small subset of patterns selected from the dictionary. The goal of dictionary learning is to simultaneously learn these patterns in the dictionary and the sparse representation of the signals [2]. Bayesian nonparametric (BNP) models based on the beta process prior represent one approach to this problem in which model selection parameters such as dictionary size are inferred directly from data [55, 71, 95, 140]. In such models, an infinite collection of beta priors constitute prior distributions on the activation probabilities of a corresponding infinite collection of dictionary elements; Bayesian nonparametric analysis ensures that a finite data set generated from a Bernoulli process will use a finite, but random number of these dictionary elements with probability one.

Typically, sparsely coding data with a dictionary entails learning which dictionary *vectors* a signal possesses [2, 71, 95]. Previous works have further used it in classifying images [4, 105, 136] and in encoding data with linear dynamical systems [63]. To a lesser extent, previous work has also been done on representing signals with latent subspaces [22, 49, 65, 77]. This generalization allows for groups of signals to be learned such that dictionary elements within the same subspace capture correlated structure within the signal. Such a representation can result in a more precise and efficient signal representation. Independent subspace analysis (ISA) [65] and mixtures of factor analyzers (MFA) [49] represent two approaches; in ISA, signals are represented as linear combinations of multiple weighted subspaces, while with MFA a signal is assigned via a mixture on subspaces to a single factor analysis model.

In the same spirit that has motivated BNP extensions to factor analysis, MFA and ISA are restricted by the fact that the number of subspaces and the dimensionality of each subspace must be defined in advance. In this work we aim to address these shortcomings using BNP priors. Specifically, we propose a beta-Bernoulli process for sparsely coding signals via *subspaces*, while we define gamma process priors on the variances of the Gaussian priors on dimensions in each subspace; posterior inference of the first prior learns the number of subspaces, while inference for the second prior learns the dimensionality of each subspace. We observe that in our model definition, each subspace can have a different dimensionality, which has a significant advantage of side-stepping the combinatorial and local-optima problem this would present for cross-validation.

We refer to our proposed model as *beta process subspace analysis* (BPSA). We develop a new scalable EM-based algorithm along the lines of other scalable dictionary learning approaches [87, 114, 115]. Our scalable algorithm can be viewed as a special case of stochastic variational inference [19, 61]. We show that BPSA is a competitive method for nonparametric dictionary learning on a denoising problem [115, 140].

3.2 Beta Process Subspace Analysis

We propose beta process subspace analysis (BPSA) for nonparametric dictionary learning. We assume that we have a set of signals $\mathbf{x} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$, where $\mathbf{x}_n \in \mathbb{R}^d$. We model these vectors as Gaussian random variables in which the mean vectors are represented hierarchically as follows. For fixed and large integer values K and D, first generate global variables

$$\mathbf{w}_k^i \,|\, \eta_k^i \;\sim\; N(0, \eta_k^i I_d), \quad \eta_k^i \;\sim\; \text{gamma}(\frac{\delta}{D}, 1),$$

$$\pi_k \;\sim\; \text{beta}\left(\alpha\frac{\gamma}{K}, \alpha(1-\frac{\gamma}{K})\right). \tag{3.1}$$

The index values are $i = 1, \ldots, D$ and $k = 1, \ldots, K$. Parameters $\alpha, \gamma, \delta > 0$ are positive and set such that $\alpha\gamma \ll K$ and $\delta \ll D$. The vector $\mathbf{w}_k^i \in \mathbb{R}^d$ corresponds to the ith dimension vector of the kth subspace and I_d indicates a d-dimensional identity matrix. In principal, we can let $K, D \to \infty$, but for inference purposes we let them be finite, but large integer values. Let $\mathbf{W}_k$ be the set of all $\mathbf{w}_k^i$ for a particular k organized in a $d \times D$ matrix. In the limit $K \to \infty$ the random measure $H_K = \sum_{k=1}^K \pi_k \delta_{\mathbf{W}_k}$ constructed from these random variables converges to a beta process, and the larger the value of K the more accurate the approximation. Similarly, the random measure $G_D^k = \sum_{i=1}^D \eta_k^i \delta_{\mathbf{w}_k^i}$ converges to a marked Poisson process as $D \to \infty$ with the measures following a gamma process. We note that a well known property of these two priors is that only a small number of $\mathbf{W}_k$ will have $\pi_k > \epsilon$ for all $\epsilon > 0$, and similarly only a small number of $\eta_k^i > \epsilon$ for a fixed k. We make a more precise statement about these asymptotics for BPSA in Proposition 3.1 below.

For each patch $\mathbf{x}_n$, sparse coding then proceeds as follows. First, independently generate

$$z_{nk} \sim \text{Bernoulli}(\pi_k), \quad c_n^k \;\sim\; N(0, I_D). \tag{3.2}$$

Then, the nth observation $\mathbf{x}_n$ is drawn

$$\mathbf{x}_n \sim N\Big(\sum_{k=1}^{K} z_{nk}(\mathbf{W}_k c_n^k)\,,\, \sigma^2 I_d\Big). \tag{3.3}$$

Sparsity dictionary learning is enforced by the beta prior on each π_k. This prior on π_k encourages $z_{nk} = 0$ for each n over all but a small number of values of k. Sparse coding results from the values of k for which π_k is large, but still allows for factors to turn on and off according to the Bernoulli process. Though the limit as $K, D \to \infty$ is a doubly-infinite sum of weighted vectors, the following proposition shows that $\mathbf{x}_n$ has finite magnitude almost surely.

Proposition 3.1. *For a vector $\mathbf{x}_n$ generated by BPSA, $\|\mathbf{x}_n\|_2 < \infty$ almost surely as $K, D \to \infty$.*

Proof 3.1. *We analyze this in the known beta and gamma process limits, rather than asymptotically. We use the facts that $\mathbb{E}[\|\mathbf{x}_n\|_2^2] < \infty$ implies $\mathbb{E}[\|\mathbf{x}_n\|_2] < \infty$, and for a vector $v \sim N(\mu, \Sigma)$, the expected squared norm is $\mathbb{E}[\|v\|_2^2] = \mu^T\mu + \mathrm{trace}(\Sigma)$. We define the sets $\mathcal{F}_1 = \{\boldsymbol{W}, \boldsymbol{z}_n, \boldsymbol{c}_n\}$ and $\mathcal{F}_2 = \{\boldsymbol{\pi}, \boldsymbol{\eta}\}$. Suppressing some conditioning on the RHS, by the tower property we have*

$$\begin{aligned}
\mathbb{E}[\|\mathbf{x}_n\|_2^2] &= \mathbb{E}[\mathbb{E}[\mathbb{E}[\|\mathbf{x}_n\|_2^2|\mathcal{F}_1]\mathcal{F}_2]] \qquad (3.4)\\
&= \mathbb{E}\left[\| \textstyle\sum_{k=1}^{\infty} z_{nk}\left(\sum_{i=1}^{\infty} c_{ni}^k \mathbf{w}_k^i\right) \|_2^2\right] + d\sigma^2\\
&= \textstyle\sum_{k=1}^{\infty} \mathbb{E}[z_{nk}]\left(\sum_{i=1}^{\infty} \mathbb{E}[(c_{ni}^k)^2]\mathbb{E}[\|\mathbf{w}_k^i\|_2^2]\right) + d\sigma^2\\
&= \textstyle\sum_{k=1}^{\infty} \mathbb{E}[\pi_k]\left(d\sum_{i=1}^{\infty} \mathbb{E}[\eta_k^i]\right) + d\sigma^2\\
&= d(\delta\gamma + \sigma^2).
\end{aligned}$$

The value $\sum_{i=1}^{\infty} \mathbb{E}[\eta_k^i] = \delta$ is the expected total measure of a gamma process with parameters δ and 1, while $\sum_{k=1}^{\infty} \mathbb{E}[\pi_k] = \gamma$ is the expected total measure of a beta process with parameters α and γ. The result follows from Chebyshev's inequality.

3.3 Example: Tiny Images

We show an illustrative example of what BPSA learns on the 80 million tiny images dataset [126]. Each color image is $32 \times 32 \times 3$ in size, giving a vectorized dimensionality of $d = 3072$. For this problem we initialized the model to $K = 100$ subspaces, each of $D = 10$ dimensions. We randomly initialized the dictionary and ran the scalable inference algorithm described in Section 3.4. In this experiment we were able to process almost of of the images with the algorithm seeing each of these images one time. The algorithm inferred 24 subspaces in this time, pruning away the remaining 76. The number of learned vectors inside each subspace varies between 1 to 9.

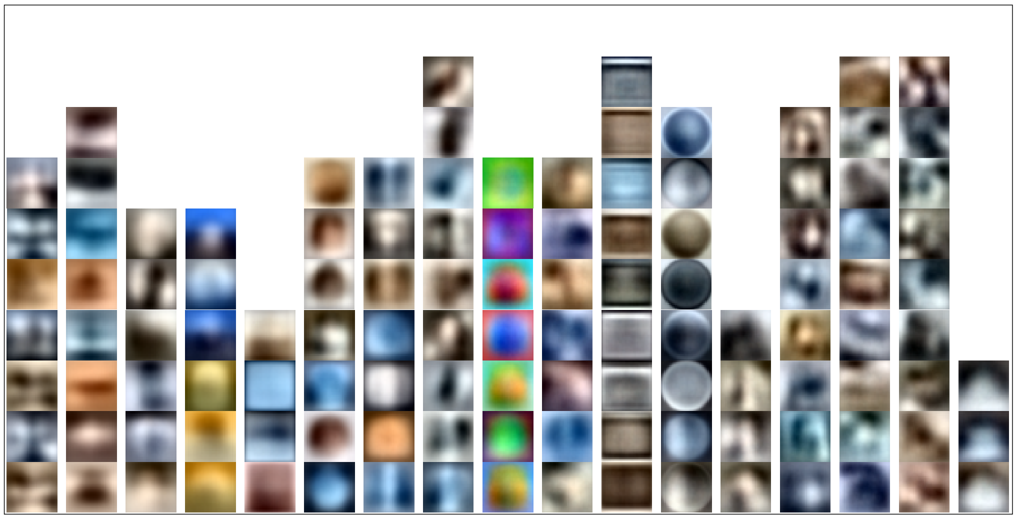


Figure 3.1: Final dictionary learned on Tiny images dataset. Each column represents a subspace and each learned vector of a subspace is reshaped to be shown by an image. From left to right, subspaces are ordered to show the most used to the least used ones in image representation.

Figure 3.1 shows the final learned dictionary. Each column shows a dictionary subspace, and vectors in subspaces are shown as $32 \times 32 \times 3$ images. We show the 17 most-used subspaces in this figure (24 were learned in total). These subspaces are shown ordered by their probability of usage from left to right. We scale the vectors of each subspace to better visualize see the learned patterns inside each subspace. It can be observed that our nonparametric model has learned only a portion of the vectors inside each subspace and pruned away the remaining.

Figure 3.2 shows the ordered probability of using each dictionary subspaces. We notice that only 24 subspaces are learned, while the remaining 76 subspaces have been pruned out by virtue of their having zero probability (i.e., not being used by data).

Finally, we randomly selected 100,000 images to investigate how many dictionary subspaces they use in their final representation. In Figure 3.3 we show a histogram of the number of used

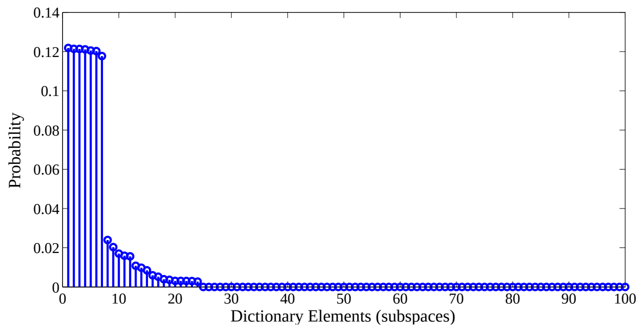


Figure 3.2: Probabilities of using each dictionary subspace in representing images ordered form the most probable to the least.

subspaces for representing particular images. As seen, almost 60,000 images used 8 subspaces in their representation, with all images using less than 9 total subspaces. Simpler images required significantly fewer subspaces. This initial experiment supports our goal in developing BPSA to nonparametrically and simultaneously learn a sparse set of varying-dimensional subspaces for data representation.

3.4 Inference for BPSA

We derive a MAP-EM algorithm for BPSA that uses a marginalization trick similar to MFA, with the result being similar to the sparse coding algorithm used by K-SVD. We then present a scalable approach for inference using stochastic optimization. We learn point estimates for each $\mathbf{W}_k$ and $\mathbf{z}_n$, and conditional posterior q distributions on all other model variables. The conditional independence induced by $\mathbf{W}_k$ and $\mathbf{z}_n$ is what makes this an exact EM algorithm, rather than a mean-field variational approximation using delta-function q distributions on $\mathbf{W}_k$ and $\mathbf{z}_n$ (see discussion below).

Let $\mathbf{x}$, $\mathbf{z}$, $\mathbf{c}$, π, η and $\mathbf{W}$ indicate sets of all of the respective variables and data. In EM for BPSA, the goal is to maximize $p(\mathbf{x}, \mathbf{z}, \mathbf{W})$ over the sparse coding vectors $\mathbf{z}_n$ and subspaces $\mathbf{W}_k$ with $\mathbf{c}$, π and η treated as marginalized variables. To this end, we define q distributions on these

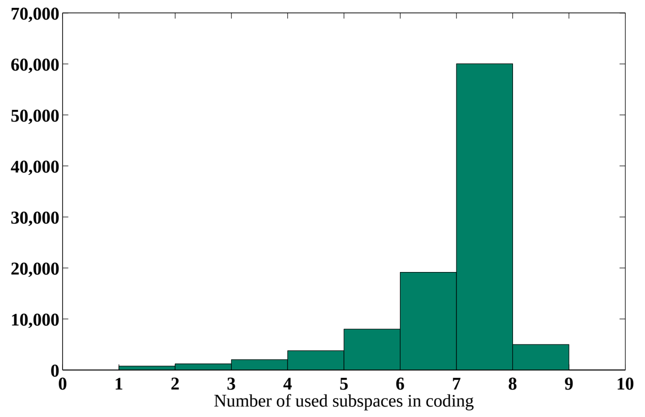


Figure 3.3: A histogram of number of subspaces used in representing 100,000 randomly selected images.

hidden variables and set up the EM objective function on the log marginal distribution,

$$\ln p(\mathbf{x}, \mathbf{z}, \mathbf{W}) \;=\; \underbrace{\mathbb{E}_q\Big[\ln \frac{p(\mathbf{x}, \mathbf{z}, \mathbf{W}, \mathbf{c}, \pi, \eta)}{q(\mathbf{c}, \pi, \eta)}\Big]}_{\mathcal{L}(\mathbf{z}, \mathbf{W})} \;+\; \underbrace{\mathbb{E}_q\Big[\ln \frac{q(\mathbf{c}, \pi, \eta)}{p(\mathbf{c}, \pi, \eta | \mathbf{x}, \mathbf{z}, \mathbf{W})}\Big]}_{\mathrm{KL}(q\|p)}. \tag{3.5}$$

The conditional posterior distribution factorizes nicely, and so we have an exact forms for $q(\mathbf{c}, \pi, \eta)$ that mirrors the factorization

$$\underbrace{p(\mathbf{c}, \pi, \eta | \mathbf{x}, \mathbf{z}, \mathbf{W})}_{q(\mathbf{c}, \pi, \eta)} = \big[\textstyle\prod_{n=1}^N \underbrace{p(\mathbf{c}_n | \mathbf{x}_n, \mathbf{z}_n, \mathbf{W})}_{q(\mathbf{c}_n)}\big] \times \big[\textstyle\prod_{k=1}^K \underbrace{p(\pi_k | \mathbf{z})}_{q(\pi_k)}\big]\big[\textstyle\prod_{k=1}^K \prod_{d=1}^D \underbrace{p(\eta_k^i | \mathbf{w}_k^i)}_{q(\eta_k^i)}\big]. \tag{3.6}$$

All three of the distributions above are in closed form and are in the Gaussian, beta and generalized inverse Gaussian families, respectively. We can update each factorized q in this way to locally optimize Eq. (3.5) over $\mathbf{z}$ and $\mathbf{W}$.

3.4.0.0.1 Connection to variational inference. We can re-frame the MAP-EM objective function of Eq. (3.5) as variational inference by using delta-function q distributions on $\mathbf{z}$ and $\mathbf{W}$. In this case, defining the factorization $q(\mathbf{c}, \pi, \eta, \mathbf{z}, \mathbf{W}) = q(\mathbf{c}, \pi, \eta)\delta_{\mathbf{z}}\delta_{\mathbf{W}}$ and writing $p(\mathbf{c}, \pi, \eta, \mathbf{z}, \mathbf{W}|\mathbf{x}) = p(\mathbf{c}, \pi, \eta|\mathbf{x}, \mathbf{z}, \mathbf{W})p(\mathbf{z}, \mathbf{W}|\mathbf{x})$, we can derive a variational inference algorithm that is identical to the MAP-EM algorithm described below. The choice $q(\mathbf{W}) = \delta_{\mathbf{W}}$ is reasonable because the posterior is tied to the data size; with large data sets, or even large individual images, a Gaussian $q(\mathbf{W})$ would be a highly peaked distribution. The other natural choice for $q(\mathbf{z})$ is a set of Bernoulli distributions, as used in [95,115], but since this effectively acts as a second weight on the dictionary elements, it can lead to scaling issues when combined with $q(\mathbf{c})$. We therefore believe that 0–1 sparse coding for $\mathbf{z}_n$ and allowing $q(c_n)$ to capture all weight information is more appropriate, and so $q(\mathbf{z}) = \delta_{\mathbf{z}}$ is a good choice.

3.4.1 Sparse coding EM step

We break the algorithm into two parts: In the first part, we derive a greedy algorithm for jointly learning $\mathbf{z}_n$ and $q(\mathbf{c}_n)$. This sparse coding step is not as straightforward as it might appear, since if $z_{nk} = 0$, then the corresponding updated q distribution on c_n^k will revert to the zero-mean prior, which makes it effectively impossible to set $z_{nk} = 1$ in the following iteration after computing the E-step over c_n^k. To mitigate this, we can integrate out $\mathbf{c}_n$ when learning $\mathbf{z}_n$. We first observe that $\ln p(\mathbf{x}, \mathbf{z}, \mathbf{W})$ can be directly optimized greedily over $\mathbf{z}$, which would be one approach. However, updating $\mathbf{W}$ is not in closed form here. Since Eq. (3.5) is an equality, one solution is to iteratively (i) update the LHS of (3.5) over $\mathbf{z}$, and then (ii) update the RHS of (3.5) over $\mathbf{W}$ and all q distributions.

We define a similar greedy algorithm, where instead of fully optimizing each $\mathbf{z}_n$ on the LHS of (3.5) and then updating the full $q(\mathbf{c}_n)$ on the RHS, we construct a *sequence* of EM equalities. Since the joint likelihood factorizes over $\mathbf{x}_n$ the objective sums over each data point and so we can sparsely code each observation independently. We can marginalize out arbitrary subsets of dimensions of $\mathbf{c}_n$ to equivalently write

$$\ln p(\mathbf{x}_n, \mathbf{z}_n|\mathbf{W}) = \mathbb{E}_q\Big[\ln \frac{p(\mathbf{x}_n, \mathbf{z}_n, \mathbf{c}_{n\mathcal{A}}, \pi|\mathbf{W})}{p(\mathbf{c}_{n\mathcal{A}}|\mathbf{x}_n, \mathbf{z}_n, \mathbf{W})p(\pi|\mathbf{z})}\Big] = \mathbb{E}_q\Big[\ln \frac{p(\mathbf{x}_n, \mathbf{z}_n, \mathbf{c}_{n\mathcal{A}}, c_n^j, \pi|\mathbf{W})}{p(\mathbf{c}_{n\mathcal{A}}, c_n^j|\mathbf{x}_n, \mathbf{z}_n, \mathbf{W})p(\pi|\mathbf{z})}\Big], \quad (3.7)$$

for a particular observation $\mathbf{x}_n$ and current sparse coding vector $\mathbf{z}_n$ and using the following definitions: $\mathcal{A} \subset \{1, 2, ..., K\}$, $j \notin \mathcal{A}$ and $\mathbf{c}_{n\mathcal{A}}$ denotes the subset of $\mathbf{c}_n$ corresponding to subspaces indexed by $\mathcal{A}$. We have also compressed the EM equality of Eq. (3.5) on the RHS side for space.

What the equality in Eq. (3.7) shows is that we can arbitrarily integrate out portions of the vector $\mathbf{c}_n$ corresponding to subspaces for which $z_{nk} = 0$. Let $\mathcal{A} = \{k : z_{nk} = 1\}$, the set of

Algorithm 3 Sparse coding greedy EM algorithm

```
1: input: Dictionary $\mathbf{W}$ and $q(\pi_k) = p(\pi_k|\mathbf{z}_1, \dots, \mathbf{z}_N)$.
2: output: Sparse coding $\mathbf{z}_n$ and $q(\mathbf{c}_n)$ (index ignored)
3: for each patch $x$ do
4:     Set $z = 0$ and index set $\mathcal{A} = \emptyset$
5:     For all $j$, initialize
           $\xi_j^+ = \ln p(x|\mathbf{W}, z_j = 1) + \mathbb{E}_q[\ln \pi_j]$
           $\xi_j^- = \ln p(x|\mathbf{W}, z_j = 0) + \mathbb{E}_q[\ln(1 - \pi_j)]$
6:     while $\max_j \; \xi_j^+ - \xi_j^- \; > \; 0$ do
7:         Set $j' = \arg\max_j \; \xi_j^+ - \xi_j^-$ (see Eq. (3.9))
8:         Augment $\mathcal{A} \leftarrow \mathcal{A} \cup \{j'\}$. Set $z_{j'} = 1$ and $\xi_j^+ = -\infty$
9:         Update $q(c_{\mathcal{A}}) = p(c_{\mathcal{A}}|x, z, W)$ (see Eq. (3.10))
10:        For all $j \notin \mathcal{A}$, update
               $\xi_j^+ = \mathbb{E}_q[\ln p(x|\mathbf{c}_{\mathcal{A}}, \mathbf{W}, z_j = 1)] + \mathbb{E}_q[\ln \pi_j]$,
               $\xi_j^- = \mathbb{E}_q[\ln p(x|\mathbf{c}_{\mathcal{A}}, \mathbf{W}, z_j = 0)] + \mathbb{E}_q[\ln(1 - \pi_j)]$
11:    end while
12: end for
```

active subspaces for observation n. (We will ignore the index n from now on.) Then our two step greedy procedure (i) calculates the marginal log likelihood in Eq. (3.7) using the equality in the first row, picking the subspace with index j that increases this value the most, and then (ii) increments the set $\mathcal{A}$ by adding index j, sets the corresponding dimension of $\mathbf{z}$ to one and recomputes the log marginal likelihood using the equality on the second row of Eq. (3.7). This augmented set then is redefined to be the first row and the procedure continues by expanding over new dimensions of $\mathbf{c}_n$. If no subspace increases the marginal likelihood, then sparse coding terminates. In this way, the Bayesian approach provides an automatic means for determining the number of subspaces appropriate for each observation.

The outline of the greedy sparse coding algorithm is given in Algorithm 1. Using the sequence of equalities constructed as in Eq. (3.7), this algorithm can be shown to monotonically increase the objective in Eq. (3.5) using the standard EM proof.

3.4.1.0.2 Procedure: As mentioned, each step of the sparse coding algorithm consists of two parts: determining which new subspace to add (or terminating) and then recomputing the log marginal likelihood using a new latent variable expansion via EM. To determine which subspace to add, we compute as a score the amount of increase in the objective function in Eq. (3.5) from adding each potential subspace. In Algorithm 1 we refer to this score as $\xi_j^+ - \xi_j^-$, where ξ_j^+ is the objective function using subspace j and ξ_j^- not using it. Again suppressing observation index n, the likelihoods used in this calculation are

$$\begin{aligned} p(\mathbf{x}|\mathbf{c}_{\mathcal{A}}, \mathbf{W}, z_j = 1) &= N(\mathbf{x}| \textstyle\sum_{k \in \mathcal{A}} \mathbf{W}_k c^k, \sigma^2 I + \mathbf{W}_j \mathbf{W}_j^T), \\ p(\mathbf{x}|\mathbf{c}_{\mathcal{A}}, \mathbf{W}, z_j = 0) &= N(\mathbf{x}| \textstyle\sum_{k \in \mathcal{A}} \mathbf{W}_k c^k, \sigma^2 I). \end{aligned} \tag{3.8}$$

Let $q(\mathbf{c}_{\mathcal{A}}) = N(\mathbf{c}_{\mathcal{A}}|\mu_{\mathcal{A}}, \Sigma_{\mathcal{A}})$ and define the residual of the approximation given the active set $\mathcal{A}$ to be $r_{\mathcal{A}} = x - \sum_{k \in \mathcal{A}} \mathbf{W}_k \mu^k$. Using the matrix inversion lemma and defining the stacked matrix $\mathbf{W}_{\mathcal{A}} = [\mathbf{W}_k]_{k \in \mathcal{A}}$, the score of subspace j equals

$$\begin{aligned} \xi_j^+ - \xi_j^- &= \frac{1}{2\sigma^2} r_{\mathcal{A}}^T \mathbf{W}_j (\sigma^2 I + \mathbf{W}_j^T \mathbf{W}_j)^{-1} \mathbf{W}_j^T r_{\mathcal{A}} \\ &+ \frac{1}{2\sigma^2} \mathrm{tr}\{\mathbf{W}_{\mathcal{A}}^T \mathbf{W}_j (\sigma^2 I + \mathbf{W}_j^T \mathbf{W}_j)^{-1} \mathbf{W}_j^T \mathbf{W}_{\mathcal{A}} \Sigma_{\mathcal{A}}\} \\ &- \frac{1}{2} \ln |I_d + \sigma^{-2} \mathbf{W}_j \mathbf{W}_j^T| \\ &+ \mathbb{E}_q[\ln \pi_j] \ - \ \mathbb{E}_q[\ln(1 - \pi_j)]. \end{aligned} \tag{3.9}$$

The expectations $\mathbb{E}_q[\ln \pi_j]$ and $\mathbb{E}_q[\ln(1 - \pi_j)]$ are in the next section. The parameters of $q(\mathbf{c}_{\mathcal{A}})$ are

$$\Sigma_{\mathcal{A}} = (I + \tfrac{1}{\sigma^2} \mathbf{W}_{\mathcal{A}}^T \mathbf{W}_{\mathcal{A}})^{-1}, \quad \mu_{\mathcal{A}} = \tfrac{1}{\sigma^2} \Sigma_{\mathcal{A}} \mathbf{W}_{\mathcal{A}}^T x. \tag{3.10}$$

These are of size $D|\mathcal{A}| \times D|\mathcal{A}|$ and $D|\mathcal{A}| \times 1$, respectively. Parsing Eq. (3.9) shows that the score measures how correlated subspace j is with the residual, and takes into account the prior probability of subspace j, as well as two other probabilistic factors. The running time of this algorithm is comparable to orthogonal matching pursuits.

3.4.2 Dictionary EM steps

After sparse coding the vectors $\mathbf{x}_n$ with $\mathbf{z}_n$, we run EM on $\mathbf{c}_n$, η and each $\mathbf{W}_k$, and update the q distribution for each π_k. In general, this part of the algorithm is very fast. Using statistics accumulated during the sparse coding step, it is limited by the matrix inversion in Eq. (3.13) below.

Given $\mathbf{z}_n$, we can collect all subspaces into a $d \times DK$ matrix $\mathbf{W} = [\mathbf{W}_1, \ldots, \mathbf{W}_K]$ and define the binary coding matrix $\mathbf{Z}_n = \text{diag}([z_{n1}\mathbf{1}_D, \ldots, z_{nK}\mathbf{1}_D])$, where $\mathbf{1}_D$ is a $1 \times D$ vector of ones. Then, the updates can be easily written as follows:

3.4.2.0.3 E-Step: This step entails updating $q(\mathbf{c}_n)$, $q(\eta_k^i)$ and $q(\pi_k)$, and then calculating $\mathbb{E}_q[\ln p(\mathbf{x}, \mathbf{z}, \mathbf{W}, \mathbf{c}, \pi, \eta)]$. We first calculate $q(\mathbf{c}_n)$ for the entire vector, which is $q(\mathbf{c}_n) = N(\mathbf{c}_n|\mu_n, \Sigma_n)$, where

$$\Sigma_n = (I + \tfrac{1}{\sigma^2}\mathbf{Z}_n\mathbf{W}^T\mathbf{W}\mathbf{Z}_n)^{-1}, \;\mu_n = \tfrac{1}{\sigma^2}\Sigma_n(\mathbf{W}\mathbf{Z}_n)^T\mathbf{x}_n. \tag{3.11}$$

The conditional posterior of the gamma process variance of vector $\mathbf{w}_k^i$—the ith dimension of the kth subspace—is $q(\eta_k^i) = \text{GiG}(\eta_k^i|e_k^i, f_k^i, p_k^i)$, where

$$e_k^i = 2, \quad f_k^i = \langle \mathbf{w}_k^i, \mathbf{w}_k^i \rangle, \quad p_k^i = -\tfrac{d}{2} + \tfrac{\delta}{D}. \tag{3.12}$$

For dictionary probabilities, $q(\pi_k) = \text{beta}(\pi_k|a_k, b_k)$ is updated $a_k = \frac{\alpha\gamma}{K} + \sum_n z_{nk}$ and $b_k = \alpha(1 - \frac{\gamma}{K}) + \sum_n(1 - z_{nk})$.

3.4.2.0.4 M-Step: Maximizing Eq. (3.5) over $\mathbf{W}$ gives the new dictionary

$$\mathbf{W} = \left[\sum_{n=1}^{N} \mathbf{x}_n\mu_n^T\mathbf{Z}_n\right]\left[\sigma^2\mathbb{E}_q[\eta^{-1}] + \sum_{n=1}^{N}\mathbf{Z}_n\mathbb{E}_q[\mathbf{c}_n\mathbf{c}_n^T]\mathbf{Z}_n\right]^{-1} \tag{3.13}$$

where $\mathbb{E}_q[\mathbf{c}_n\mathbf{c}_n^T] = \mu_n\mu_n^T + \Sigma_n$ and $\mathbb{E}_q[\eta^{-1}]$ is the diagonal matrix

$$\mathbb{E}_q\Big[\eta^{-1}\Big] = \begin{pmatrix} \mathbb{E}_q\big[\frac{1}{\eta_1^1}\big] & & & & & & \\ & \ddots & & & & & \\ & & \mathbb{E}_q\big[\frac{1}{\eta_1^D}\big] & & & & \\ & & & \ddots & & & \\ & & & & \mathbb{E}_q\big[\frac{1}{\eta_K^1}\big] & & \\ & & & & & \ddots & \\ & & & & & & \mathbb{E}_q\big[\frac{1}{\eta_K^D}\big] \end{pmatrix}$$

Each of the diagonal entries can be found by calculating

$$\mathbb{E}_q[(\eta_k^i)^{-1}] = \frac{\sqrt{2}}{\|\mathbf{w}_k^i\|_2}\frac{\mathcal{K}_{(-\frac{d}{2}+\frac{\delta}{D}-1)}\big(\sqrt{2}\|\mathbf{w}_k^i\|_2\big)}{\mathcal{K}_{(-\frac{d}{2}+\frac{\delta}{D})}\big(\sqrt{2}\|\mathbf{w}_k^i\|_2\big)}. \tag{3.14}$$

$\mathcal{K}$ is the modified Bessel function of the second kind.

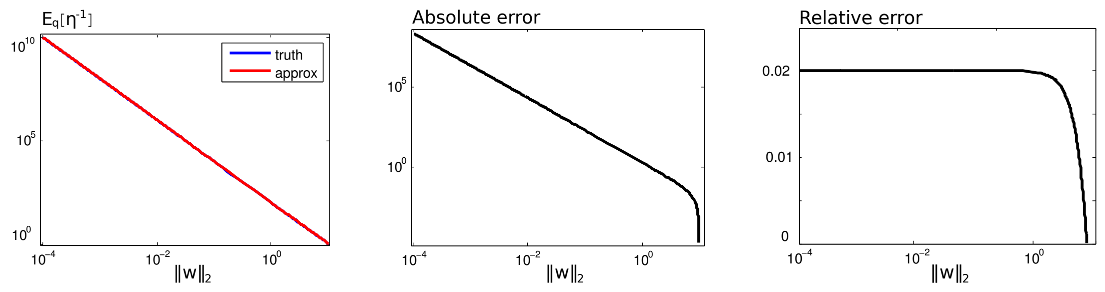


Figure 3.4: Accuracy of the Bessel approximation in Eq. (3.14) and (3.15) (with $\delta = 1$, $D = 25$, $d = 100$). (left) The true and approximate functions, (middle) the absolute error of the approximation, (right) the approximation error as a fraction of the true value.

3.4.2.0.5 Approximation: When the dimensionality of $\mathbf{w}_k^i$ is large, or as its magnitude goes to zero, calculating the numerator and denominator of Eq. (3.14) separately can lead to numerical issues. We can approximate this ratio using their asymptotic forms,

$$\begin{aligned} &\mathcal{K}_a(b) \sim \tfrac{1}{2}\Gamma(|a|)(\tfrac{1}{2}b)^{-|a|} \\ &\mathbb{E}_q[(\eta_k^i)^{-1}] \approx 2(1 + \tfrac{d}{2} - \tfrac{\delta}{D})/\|\mathbf{w}_k^i\|_2^2 \end{aligned} \tag{3.15}$$

The positive reinforcement of the shrinkage property of the gamma process is clear since, as $\|\mathbf{w}_k^i\|_2$ becomes small, the update for $\mathbf{w}_k^i$ is shrunk even more to the zero vector. We show the accuracy of this approximation in Figure 3.4. As shown in this figure, the approximation is almost exact for small norm subspace vectors in log scale. After these updates, we return to sparse coding steps to update each $\mathbf{z}_n$ and iterate until convergence.

3.4.3 Scalable inference with stochastic EM

We can also scale inference for larger data sets with stochastic EM [19]. At iteration t, let $\mathcal{L}_t$ be the corresponding portion of (3.5) restricted to $\mathbf{x}_n$ for $n \in S_t \subset \{1, \ldots, N\}$, where S_t is selected uniformly at random at each iteration, and scaled by $N/|S_t|$. We update $\mathbf{W}$ with the gradient step

$$\mathbf{W}^{(t+1)} = \mathbf{W}^{(t)} + \rho_t(-\nabla^2_{\mathbf{W}}\mathcal{L}_t)^{-1}\nabla_{\mathbf{W}}\mathcal{L}_t. \tag{3.16}$$

Here, $\rho_t > 0$, and $\sum_t \rho_t = \infty$, $\sum_t \rho_t^2 < \infty$ [16]. This produces the straightforward update

$$\begin{aligned}\mathbf{W}'_t = &\left[\sum_{n\in S_t} \mathbf{x}_n \mu_n^T \mathbf{Z}_n\right] \times \\ &\left[\sigma^2 \frac{|S_t|}{N} \mathbb{E}_q[\eta^{-1}] + \sum_{n\in S_t} \mathbf{Z}_n(\mu_n\mu_n^T + \Sigma_n)\mathbf{Z}_n\right]^{-1},\end{aligned} \tag{3.17}$$

$$\mathbf{W}^{(t+1)} = (1-\rho_t)\mathbf{W}^{(t)} + \rho_t \mathbf{W}'_t. \tag{3.18}$$

In other words, we first calculate the optimal update of $\mathbf{W}$ restricted to subset S_t, using the scaling factor $N/|S_t|$, and then take a weighted average of this update with the current value.

Stochastic optimization for the other global variables π follows exactly from the stochastic variational inference framework [61] (see, e.g., [115]). In short, to update each $q(\pi_k)$ we form the updates for a_k and b_k of $q(\pi_k)$, but limited to the set S_t and scaled appropriately [61]. Then take a ρ_t-weighted average of these values with the old values similar to the update of $\mathbf{W}$ above.

At step t, for $k = 1, ..., K$ first set

$$\begin{aligned} a'_k &= \frac{\alpha\gamma}{K} + \frac{N}{|S_t|}\sum_{n\in S_t} z_{nk}, \\ b'_k &= \alpha(1-\frac{\gamma}{K}) + \frac{N}{|S_t|}\sum_{n\in S_t}(1-z_{nk}). \end{aligned} \tag{3.19}$$

This focuses on sparse coding of the data in S_t. Then set

$$\begin{aligned} a_k^{(t+1)} &= (1-\rho_t)a_k^{(t)} + \rho_t a'_k, \\ b_k^{(t+1)} &= (1-\rho_t)b_k^{(t)} + \rho_t b'_k. \end{aligned} \tag{3.20}$$

3.5 Experiments

We compare BPSA with BPFA [95], K-SVD [2], mixtures of factor analyzers (MFA) [49] and total variation denoising [52] on denoising tasks. We observe that, with the exception of total variation, the algorithms we compare with belong to a closely related family of dictionary learning models: BPFA is a special case of BPSA with the subspace dimensionality set to one, while BPFA is a Bayesian nonparametric extension of K-SVD. MFA can be viewed as a version of

BPSA where each observation possess exactly one subspace, which is drawn from a multinomial distribution instead of a Bernoulli process. In this way, MFA can be viewed as another special case of BPSA in the other “direction” from the special case of BPFA. Total variation represents a fundamentally different modeling approach, which we include to show the advantage of dictionary learning for denoising.

3.5.1 Setup

We present experimental results on an image denoising task. We use four classic test images, “Peppers,” “House,” “Lena,” and “Barbara,” shown in Figure 3.5. To each image, we add white Gaussian noise with standard deviations $\sigma \in \{10, 15, 20, 25, 50\}$. To quantitatively assess performance, we use the Structural Similarity Index Measure (SSIM) and Peak Signal to Noise Ratio (PSNR) [134] of the denoised image to the ground truth.

For BPFA, we use the implementation of [114]. For K-SVD, we use the code provided by [2]. In all algorithms, we use the technique in [83] to set the noise parameter; for TV-minimization this provides us with the target empirical noise when setting the regularization parameter on the L1 penalty. To set the parameters for MFA, we use cross validation as suggested in the paper by [49]. Therefore, all the algorithms are compared under the same noise assumption, which we observed was close to the ground truth.

For BPSA, BPFA, K-SVD and MFA, we extracted 16×16 patches from each image using shifts of one pixel, which overall produced the best results for all algorithms compared with 8×8 and 12×12. We ran the algorithms with the following settings: For BPFA and K-SVD, we set $\eta = 255^2$ and $K = 256$, for MFA we set the mixture size to $M = 50$ and $D = 20$ to be the subspace size of each mixture, for BPSA we set $D = 20$, $K = 200$, $\delta = 0.1$, $\alpha = 1$ and $\gamma = 1$. For stochastic BPSA and stochastic BPFA, we set $|S_t| = 1000$ and use a step size $\rho_t = (t_0 + t)^{-\kappa}$, with $t_0 = 10$ and $\kappa = 0.75$.

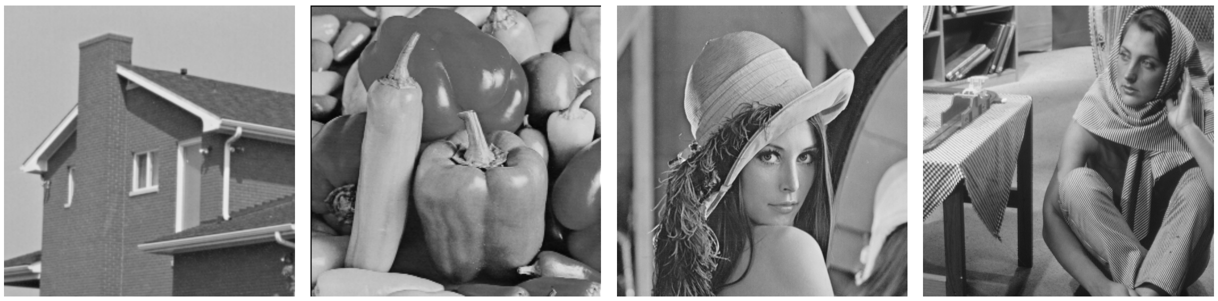


Figure 3.5: Images used in our denoising experiments: House, Peppers, Lena, Barbara.

3.5.2 Denoising results

In Table 3.1 we show example PSNR and SSIM results for denoising the images in Figure 3.5. The baseline is the noisy image. The sparse coding algorithms all perform similarly, but overall augmenting with subspaces as BPSA does improve the denoising results. We notice that MFA performs significantly worse than the dictionary learning methods, which clearly shows the advantage of sparse coding versus clustering. Total variation performs the worst of all algorithms, showing the advantage of a local optimal solution to a non-convex model that captures greater structure over a global optimal solution of a simpler, but convex model.

The dictionary learning for BPSA and BPFA took approximately 20 and 3.5 minutes, respectively, for each image, followed by one iteration over all dictionary elements for reconstruction. Although BPSA is slower than BPFA, it is able to more effectively capture the data structures that improves the performance of denoising.

We show the sparsity of the model and subspaces in Figure 3.6 for three images and $\sigma = 15$. For each image, we show the histogram for the number of learned dimensionality of each subspace in their dictionary We see that BPSA infers subspaces of varying size, including one-dimensional subspaces as learned with BPFA and K-SVD, while learning between 50 and 100 subspaces overall. Note that for each image, the number of learned subspaces in their dictionary is different. To show the sparsity in representing the data, we compare the average number of dictionary vectors used by a patch. We show these results as a function of σ for three images in Figure

Table 3.1: SSIM | PSNR for image as a function of noise standard deviation.

HOU.	$\sigma = 10$		$\sigma = 15$		$\sigma = 20$		$\sigma = 25$		$\sigma = 50$	
BPSA	**0.943**	**35.75**	**0.922**	**34.06**	**0.903**	**32.89**	**0.890**	**31.81**	**0.829**	**28.58**
BPFA	0.941	35.58	0.914	33.64	0.887	32.32	0.872	31.20	0.811	27.64
K-SVD	0.924	35.43	0.888	33.56	0.879	32.61	0.864	31.51	0.800	28.01
MFA	0.880	29.05	0.868	28.97	0.859	26.77	0.847	26.52	0.803	25.82
TV.aiso	0.874	33.76	0.847	31.89	0.831	30.76	0.817	29.91	0.753	27.04
TV.iso	0.873	33.72	0.846	31.85	0.831	30.73	0.819	29.96	0.762	27.12
Baseline	0.627	28.12	0.481	24.63	0.387	22.15	0.319	20.13	0.161	14.13
PEP.	$\sigma = 10$		$\sigma = 15$		$\sigma = 20$		$\sigma = 25$		$\sigma = 50$	
BPSA	**0.960**	32.84	**0.944**	**31.62**	**0.928**	**30.03**	**0.916**	**29.11**	**0.847**	**25.92**
BPFA	0.955	**33.00**	0.930	30.85	0.926	29.98	0.913	28.98	0.840	25.54
K-SVD	0.952	32.99	0.911	31.25	0.900	29.53	0.894	28.35	0.809	24.71
MFA	0.911	28.77	0.910	27.70	0.890	26.67	0.881	26.20	0.841	25.80
TV.aiso	0.903	32.40	0.872	30.44	0.850	29.25	0.828	28.26	0.744	25.37
TV.iso	0.905	32.56	0.875	30.59	0.853	29.42	0.832	28.40	0.751	25.48
Baseline	0.718	28.15	0.584	24.61	0.485	22.09	0.411	20.17	0.215	14.12
LENA	$\sigma = 10$		$\sigma = 15$		$\sigma = 20$		$\sigma = 25$		$\sigma = 50$	
BPSA	**0.937**	34.65	0.917	**33.04**	**0.902**	**31.78**	**0.888**	**30.93**	**0.810**	**27.85**
BPFA	**0.937**	34.64	**0.919**	32.99	0.901	31.70	0.885	30.66	0.806	27.37
K-SVD	0.932	**34.87**	0.906	33.01	0.878	31.53	0.857	30.48	0.761	26.82
MFA	0.865	29.34	0.856	28.72	0.850	27.02	0.839	26.99	0.783	26.01
TV.aiso	0.874	32.71	0.841	30.96	0.816	29.84	0.793	28.87	0.725	26.47
TV.iso	0.874	32.78	0.843	31.04	0.818	29.93	0.796	28.98	0.731	26.57
Baseline	0.646	28.14	0.493	24.61	0.390	22.12	0.318	20.19	0.145	14.14
BAR.	$\sigma = 10$		$\sigma = 15$		$\sigma = 20$		$\sigma = 25$		$\sigma = 50$	
BPSA	**0.959**	33.76	**0.945**	**32.34**	**0.928**	**30.61**	**0.911**	**29.57**	**0.816**	**26.23**
BPFA	**0.959**	33.90	0.943	32.04	0.926	30.52	0.907	29.27	0.800	25.48
K-SVD	0.956	**33.96**	0.934	31.72	0.909	30.16	0.882	28.80	0.735	24.62
MFA	0.902	29.11	0.894	26.14	0.853	24.70	0.822	24.34	0.767	24.11
TV.aiso	0.877	29.77	0.820	27.49	0.770	26.00	0.728	25.07	0.609	22.96
TV.iso	0.877	29.77	0.822	27.50	0.773	26.01	0.734	25.12	0.618	23.02
Baseline	0.739	28.13	0.614	24.63	0.518	22.13	0.444	20.18	0.227	14.15

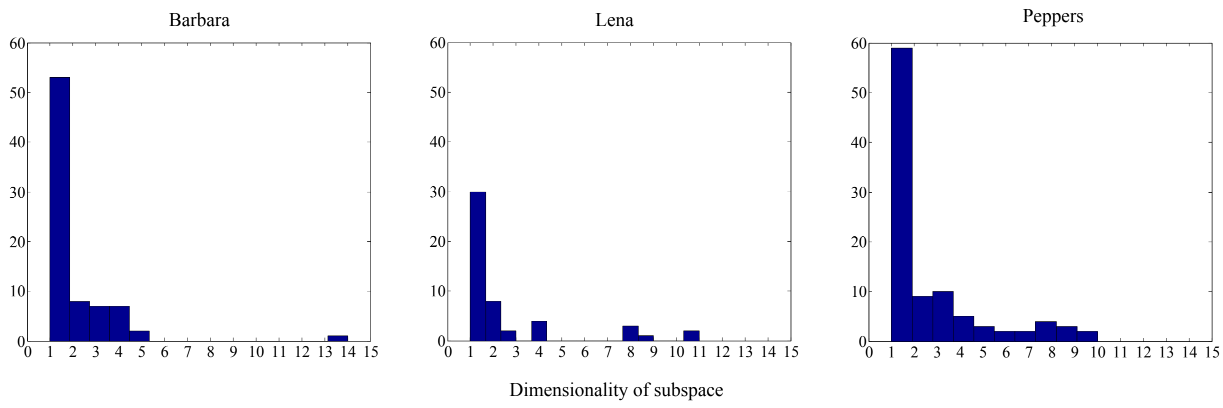


Figure 3.6: A histogram of the subspace dimensionality for $\sigma = 15$. Many inferred subspaces are one dimensional, similar to BPFA and K-SVD, but BPSA learns subspaces with other sizes as well.

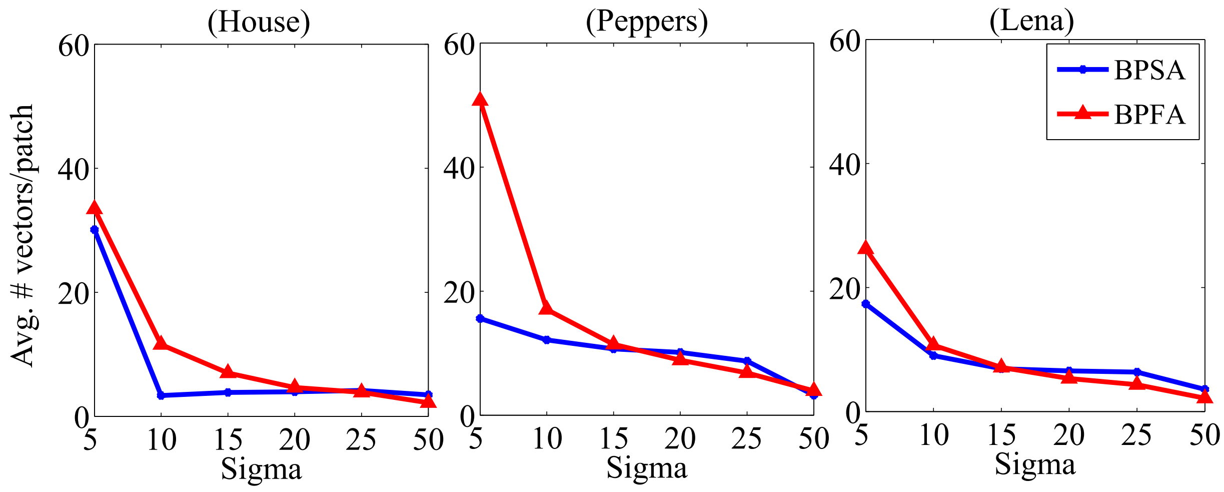


Figure 3.7: The average number of vectors used from the dictionary by a patch for various noise settings. For BPSA, since all vectors in an activated subspace are used, we calculated this number by summing the dimensionality of each subspace used by a patch.

3.7. For BPFA, we take the average number of dictionary elements used per patch. For BPSA, we first find which subspaces are used by a patch and then sum the inferred dimensionality of

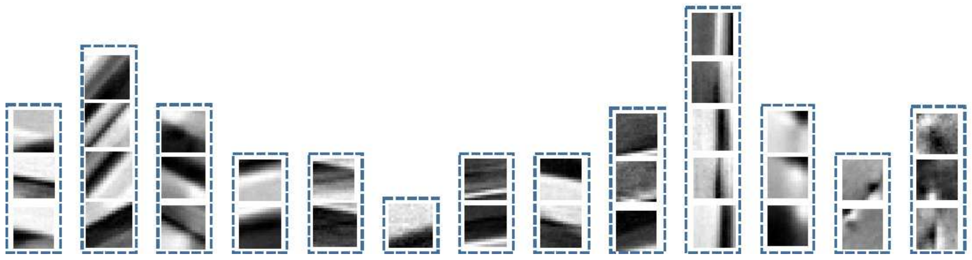


Figure 3.8: The 13 most probable subspaces learned from the "House" image with $\sigma = 15$.

each of these subspaces, since a subspace is not sparsely used in our model. As can be seen in Figure 3.7, BPSA results in a sparser signal representation (less number of used dictionary elements) as the noise decreases, and is comparable to BPFA as it increases. In Figure 3.8 we show the learned subspaces for the "House" images with $\sigma = 15$. As is evident, each subspace shares a common structure learned from the data.

3.6 Conclusion

We presented a new Bayesian nonparametric model called *beta process subspace analysis* (BPSA) for dictionary learning that sparsely codes signals in latent subspaces. The is model an extension of related methods such as BPFA and MFA. Using beta and gamma processes, it can infer both the number of subspaces and the dimensionality of each subspace. We derived a new MAP-EM based algorithm that is related to variational inference and the OMP algorithm used by K-SVD. We illustrated the model procedure on Tiny Images data set and demonstrated the advantage of sparse coding with subspaces on denoising problems.

Chapter 4

Mixed Membership Recurrent Neural Networks

Models of sequential data such as the recurrent neural network (RNN) often implicitly treat a sequence as having a fixed time interval between observations and do not account for group-level effects when multiple sequences are observed. We propose a model for grouped sequential data based on the RNN that accounts for varying time intervals between observations in a sequence by learning a group-level parameter to which each sequence reverts as more time passes between observations. Our approach is motivated by the mixed membership framework, and can be used for dynamic topic modeling-type problems in which the distribution on topics (not the topics themselves) are evolving in time. We demonstrate our approach on two datasets: The Instacart set of 3.4 million online grocery orders made by 206K customers, and a UK retail set consisting of over 500K orders [45].

4.1 Introduction

Recurrent neural networks (RNNs) are now standard models for sequential data analysis [36, 111]. Each time step of an RNN models an observation via a neural network using the observation and hidden states from previous time points. Sequential models such as the RNN (as well as the hidden Markov model and others) often implicitly assume a fixed time interval between these observations. They also often do not account for group-level variation when multiple sequences are observed, each assigned to one group. For example, consider the individual sequences of purchases by a set of customers, with one sequence per customer. A vanilla RNN implementation models these sequences with a shared network that removes customer-level information, and according to an indexing that removes the time interval information between orders. However, this information is important, since an interval of one day versus one month between orders significantly impacts the items likely to be purchased next, while modeling customer information can help inform what this impact should be.

Although some common methods exist, there is no standard technique for addressing varying time-lags in a sequence, and none that consider this with additional group-level local information. Previous work tends to simply impute the missing values with either zeros, the last observed value or the global mean of the data [7, 26, 82, 99, 127]. In [21], the authors propose a method to directly address the missingness pattern in a single data sequence by modifying a gated recurrent unit (GRU), but this is not obviously modifiable to learn local effects for groups of sequences, or to other RNN architectures such as LSTM. Group-level information was addressed for topic models by [138] through an evolving sequence of topic distributions, but the approach does not consider time lag as existing in the data. Time lag is of interest to [66] for predicting the time to the next event, but not for how a distribution on that event should evolve as a result, while [132] and [32] also do not factor user-level variation as being relevant to their formulations.

In this chapter we propose a sequential modeling approach that can be viewed as a continuous-time mixed membership RNN (Section 4.3). Our perspective is to assume that, as more time passes between any two observations, the value of the sequential information for making the next prediction decreases. To this end, each set of sequences shares the same RNN parameters as being sufficiently powerful to model user behavior, while having a local bias vector to which that group can revert as more time passes between observations (illustrated in Figure 4.1). Experiments on two datasets demonstrate the advantage of this added model flexibility (Section 4.4).

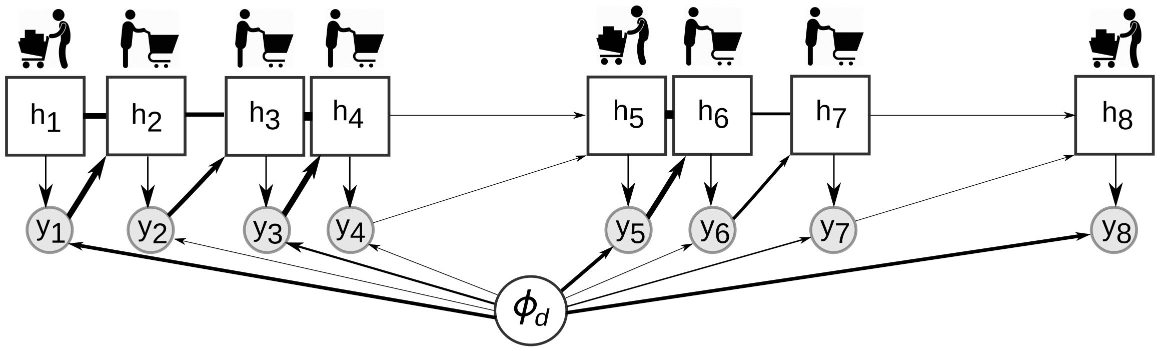


Figure 4.1: The unrolled proposed framework we use in our experiments shown for the dth sequence. Each prediction uses a weighted combination of an RNN and a customer-specific parameter. As the time between observations decreases the RNN prediction is favored more heavily. As the time lag increases the prediction is more biased towards an independent distribution parameterized by ϕ_d. (This difference is indicated by arrow thickness above.)

4.2 Background

4.2.1 Recurrent Neural Networks

A recurrent neural network models a sequence of vectors $\mathbf{y} = (y_1, \dots, y_T)$ with a neural network that takes as input a corresponding sequence of vectors $\mathbf{x} = (x_1, \dots, x_T)$ along with internal hidden states from the network, $\mathbf{h} = (h_1, \dots, h_T)$. When the model is probabilistic, this can be viewed as defining a joint likelihood of the data, $p(\mathbf{y}|\theta) = p(y_1|\theta) \prod_{t>1} p(y_t|y_{1:t-1}, \theta)$, where $p(y_t|y_{1:t-1}, \theta) \equiv p(y_t|h_t)$ and $h_t = f_\theta(x_t, h_{t-1})$ for $t > 0$, usually with $x_t \equiv y_{t-1}$. The non-linear function f_θ can be a standard RNN cell, or a more complex GRU [25] or LSTM [60], and θ are its parameters. We will use a function of the form $f_\theta(x_t, h_{t-1}) \equiv f_{\hat{\theta}}(Wx_t + Uh_{t-1})$, where W and U are matrices. Possible forms of the distribution $p(y_t|h_t)$ include Gaussian, multinomial and Poisson, as determined by the problem. The goal is typically to do maximum likelihood or MAP inference, depending on whether priors are on θ.

When multiple sequences are observed, a typical and straightforward approach is to treat them as independent from the same RNN, $p(\mathbf{y}_1, \dots, \mathbf{y}_D|\theta) = \prod_d p(\mathbf{y}_d|\theta)$. This explicitly treats each observation sequence as having the same distribution. This may be sufficient with large model capacity, but has difficulty adapting to varying time lags between observations. For example, imputation techniques can cause a sequence to revert to the same base prediction; more data-tailored methods can be useful here. To account for variations in sequences across multiple groups, mixtures of RNNs are one straightforward approach. In this work we take a different approach motivated by the mixed membership modeling framework described below.

4.2.2 Mixed membership models

Mixed membership models provide a probabilistic approach to modeling groups of data through a combination of shared and group-specific parameters [3]. The best known mixed membership model is latent Dirichlet allocation (LDA) [13], but many variations exist. The basic generative structure of a mixed membership model is:

1. Generate global variables $\theta \sim p(\theta)$ shared by all groups of data.
2. For the dth group of data: Generate local variables $\phi_d \sim p(\phi)$, and data $y_d \sim p(y_d|\phi_d, \theta)$.

The distribution $p(y_d|\phi_d, \theta)$ is a mixture where the variables θ define the globally shared set of distributions and ϕ_d is used to define the weights on these distributions.

LDA and related models let $\theta = \{\beta_1, \dots, \beta_k\}$ be a set of distributions on a discrete item set (often words in a vocabulary), and ϕ_d be a probability vector on those topics. One example closely related to our work is the correlated topic model (CTM) [12], in which $\phi_d \sim \mathcal{N}(\mu, \Sigma)$, and a softmax function $\sigma(\phi_d)$ transforms this vector into a distribution on topics. A key benefit of such models is that each group of data can mix over the same set of distributions, allowing them to share statistical strength during inference, while also allowing a meaningful comparison across groups via their shared representation in these distributions. In the next section, we are motivated by this mixed membership modeling perspective when defining a shared RNN for multiple sequences that also allows each sequence to have its own unique characteristics.

Algorithm 4 Basic MM-RNN

```
 1: Define $\Delta t$ be time since last observation at
 2:    time $t$ and $\rho(\Delta t) \in [0, 1]$ decreasing in $\Delta t$.
 3: Given RNN cell $f_\theta$ and nonlinearity $\sigma$:
 4: Generate RNN parameters $\theta \sim p(\theta)$
 5: Generate group vectors $\phi_d \sim \mathcal{N}(\mu, \Sigma)$
 6: for $d$th group sequence $\mathbf{y}_d(t)$ do
 7:     Compute $h_t = f_\theta(x_t, h_{t-1})$
 8:     Compute $\sigma_t = \sigma(\rho(\Delta t)h_t + (1 - \rho(\Delta t))\phi_d)$
 9:     Generate $\boldsymbol{y}_d(t) \sim p(y|\sigma_t)$
10: end for
```

Algorithm 5 MM-RNN topic model

```
 1: After first 5 lines of Algorithm 1, make
 2:     the additional modifications:
 3: Generate topics $\beta_k \sim \mathrm{Dir}(\alpha),\ k \in [K]$
 4: for $d$th group sequence $\mathbf{y}_d(t)$ do
 5:     Compute $h_t = f_\theta(x_t, h_{t-1})$
 6:     Compute topic distribution vector
 7:       $\sigma_t = \mathrm{softmax}(\rho(\Delta t)h_t + (1 - \rho(\Delta t))\phi_d)$
 8:     Generate $\boldsymbol{y}_d(t)$ according to a mixture of
 9:         multinomial parameters, $\sum_d \sigma_{t,k}\delta_{\beta_k}$
10: end for
```

4.3 Mixed Membership RNN Models

Recurrent neural networks work well on sequential data, but have difficulty capturing global semantic information. By contrast, topic models have the ability to capture global semantics, but are usually not sequential models and lack the modeling power of the RNN in this regard. Recent work by [30] has demonstrated the advantage of combining these two modeling paradigms for natural language models of text, which motivates our significantly different approach to modeling grouped count sequences.

As mentioned, we are motivated by the sequence modeling problem in which a long delay between observations results in the loss of value of the previous sequential information for predicting the next observation. To this end, we propose a model that accounts for the following: 1) each prediction within a group's sequence is influenced by both the previous sequential information and biases that are group-specific; 2) as the time intervals increase, the prediction smoothly adapts toward the base prediction and away from what the purely sequential prediction of the RNN would be. For example, in a dynamic topic modeling problem in which the topics are fixed and a sequence of topic distributions are generated, the topic distribution smoothly reverts to a group-specific base topic distribution as the time between documents increases.

4.3.1 The basic framework

We first present the basic idea of the model directly on data $\mathbf{y}_d$, $d = 1, \ldots, D$, where each $\mathbf{y}_d$ is a sequence of vectors with corresponding sequence of time stamps. In this model, we define $\rho(\Delta t) \in [0, 1]$ to be a function of the time interval between two particular observations in a sequence, Δt. This value produces a weighted average and decreases as Δt increases. For example, in our experiments we use $\rho(\Delta t) = (t_0 + \Delta t)^{-\kappa}$ with $t_0, \kappa > 0$. $\rho(\cdot)$ will allow us to define a continuous-time RNN that adjusts to periods of no observations.

The basic MM-RNN model is shown in Algorithm 1. To give two specific examples, if y_t were a histogram of counts, then σ_t could be the softmax function and $p(y|\sigma)$ a multinomial leading to the cross entropy penalty. Or $p(y|\sigma)$ could be a (technically inappropriate) Gaussian distribution on the normalized y with σ as the mean, resulting in an L2 penalty. In the proposed framework, we modify the RNN by including a group-specific bias vector $\phi_d \in \mathbb{R}^K$. Then, rather than generate $\mathbf{y}_d(t)$ dependent on $\mathbf{h}_d(t)$ as in the typical RNN setup, in Step 3(b) we average $\mathbf{h}_d(t)$ with ϕ_d according to the function ρ. As discussed, ρ decreases as the time interval between $\mathbf{y}_d(t-1)$ and $\mathbf{y}_d(t)$ increases. When $\rho = 0$, $\mathbf{y}_d(t)$ is independently generated from the base distribution for group d. The definition of $\rho(\cdot)$ determines the rate at which the RNN *is forgotten*;

the RNN can have its own forgetting mechanism as well. When $\rho = 1$ the sequence is being fully modeled by an RNN. We show the basic graphical model of our network in Figure 4.2.

We anticipate that this approach can give better predictions by: 1) not artificially learning sequential information that it isn't there, and 2) allowing a better RNN to be learned by focusing on the part of the data where sequential information is present, which we consider to be when the time between observations is short.

4.3.2 A mixed membership RNN topic model

We extend the basic MM-RNN idea to address the topic modeling problem. Topic models capture semantic meaning through a mixture of K topics $\boldsymbol{\beta} = \{\beta_1, \ldots, \beta_K\}$, being probability distributions on a vocabulary of size V. Each document is a set of words generated using a K-dimensional mixing weight vector on these topics, $\sigma^{(d)}$ for document d. A document $y^{(d)}$ consists of n_d words, where for each word instance a topic index is chosen according to $\sigma^{(d)}$ and the word value is then chosen by drawing from the distribution in $\boldsymbol{\beta}$ with that index. The topics learned are semantically meaningful, and topic models are powerful in that they can be used for far more than text data.

The canonical topic model for sequential data is the dynamic topic model (DTM) [11]. There, the topics vary in time, while each document generates its own $\sigma^{(d)}$ independently and uses the snapshot of topics at the moment of its generation. This allows prominent words within a coherent topic (e.g., the "politics" topic) to evolve over time. Here we consider a different problem where the topics are fixed in time, and the distributions on topics evolve. For example shopping behavior data consists of products (words) in an order (document), and each customer's sequence of orders can be modeled by a mixed membership model where each order's distribution on a fixed set of topics evolves over time.

We describe our general MM-RNN topic model in Algorithm 2. The data-generating distribution in Step 4(c) can be the standard mixture of multinomials used by LDA, or it could be a Poisson matrix factorization, or other distribution on count data. To connect this with previous topic models, we observe that if $\rho \equiv 0$ and each group consists of one "document," then this

Algorithm 3 MM-RNN learning outline

```
Initialize RNN parameters $\theta$ and initialize all $\phi_d = 0$.
Iterate the following:
    1. Update each $\phi_d$ via gradient descent
    2. Update RNN $\theta$ via automatic differentiation
    3. (optional) Update topic matrix via multiplicative update. (Otherwise fix $B = I$.)
```

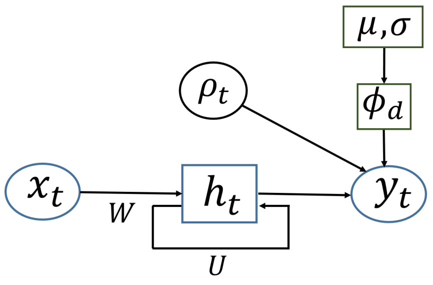


Figure 4.2: MM-RNN graphical model.

model reduces to the correlated topic model (CTM) [12]. In this sense the proposed model is one possible version of a dynamic CTM.

4.3.3 Discussion on model inference

We have presented our MM-RNN approach in fairly general terms. In this section we discuss two possible instances that we consider in our experiments and discuss an outline of how we optimized them. We discuss MAP optimization for these models.

In our models, we let f_θ, used to construct the hidden state h, be a single layer LSTM cell as is standard in PyTorch. Let $y_{d,t}$ be a probability vector or histogram, for example constructed from items purchased in order t by customer d. Using zero-mean Gaussian priors on all model variables, we can write one possible objective function as

$$\mathcal{L} = \frac{1}{2a}\|\theta\|^2 + \sum_{d=1}^{D} \frac{1}{2b}\|\phi_d\|^2 + \sum_{t=1}^{T_d} \mathcal{L}(y_{d,t}, v_{d,t}) \tag{4.1}$$

where

$$v_{d,t} \equiv \rho_{d,t} h_{d,t} + (1 - \rho_{d,t})\phi_d,$$

and again, $\rho_{d,t}$ is a deterministic, decreasing function of the time between orders ($\rho_{d,1} = 0$). Depending on the model that we choose, the loss function $\mathcal{L}(.)$ could be an average cross entropy loss $\frac{1}{T_d}\text{CEL}(\mathbf{y}_{d,t}, \sigma(v_{d,t}))$ or squared norm error $\frac{1}{2c}\|y_{d,t} - B\sigma(v_{d,t})\|^2$.

We give a rough outline of what the learning algorithm looks like in Algorithm 3. We note here that we take the perspective of nonnegative matrix factorization (NMF) using the L2 penalty when the matrix of "topics" B is incorporated. In this case, we are doing maximum likelihood on B and the columns do not need to sum to one, yet are still interpretable. B can be learned using the simple multiplicative update strategy of [78] as follows,

$$B_{i,j} \leftarrow B_{i,j} \frac{(\mathbf{y}\sigma^T)_{i,j}}{(B\sigma\sigma^T)_{i,j}}, \tag{4.2}$$

where $\mathbf{y}$ and σ show matrix of all target vectors and RNN outputs, respectively.

4.4 Experiments

We experiment with two data sets described in the following sections. We first perform a more detailed quantitative evaluation of the advantage of our time-adaptive MM-RNN approach compared with purely sequential and purely i.i.d. approaches. This is followed by a comparative quantitative evaluation against other possible approaches.

4.4.1 Instacart online grocery shopping dataset

In this section, we present experiments on the Instacart 2017 online grocery shopping data set.[1] This data consists of 3.4 million orders made by 206K users. The time interval between orders is number of days (capped at 30 days). Each order consists of a count of the number of each product purchased from 50K products and each product belongs to one of 134 aisles. In our experiments, we consider the basic MM-RNN model at the aggregated aisle level, and the MM-RNN topic model at the product level. We train all models on the orders of all customers except for the last order of each customer, which we hold out for prediction to evaluate performance.

We implement our models in PyTorch using automatic differentiation [100] and stochastic gradient descent with a learning rate of 0.01. For our selected RNN, we an LSTM with hidden dimension of 10. When $\rho \equiv 1$, our MM-RNN reverts to this LSTM, which is one of the models

[1]`https://instacart.com/datasets/grocery-shopping-2017`

we compare with. Experiments are done on a cluster node with two NVIDIA Tesla K80 GPUs and 128 GB memory.

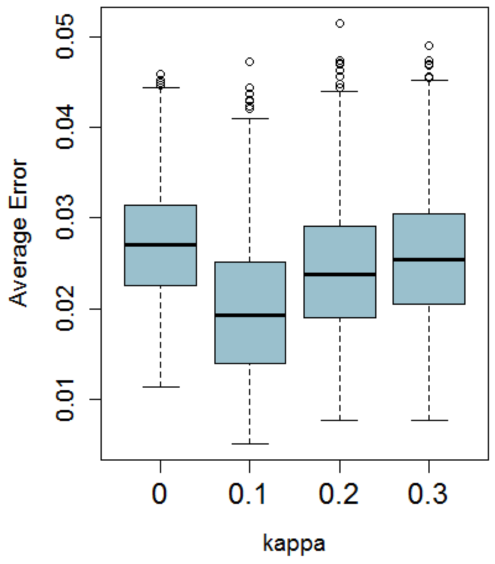
4.4.1.1 Aisle level model

In our first experiment, we consider the basic MM-RNN model of Section 4.3.1 on Instacart data aggregated at the aisle level as defined by this online shopping website (e.g., coffee, milk, cereal, tofu meat alternatives—134 aisles in total). Each order is represented as a normalized histogram giving an empirical distribution of that order across the aisles. We use the softmax function for σ to predict this distribution for the next order in the sequence. Using the function $\rho(\Delta t) = (t_0 + \Delta t)^{-\kappa}$, we set $t_0 = 1$ and experiment with various values of κ. For each experiment, we learned the model by running 20 epochs over the data, where each epoch took approximately 5 minutes. For each setting we ran 50 experiments with random initialization.

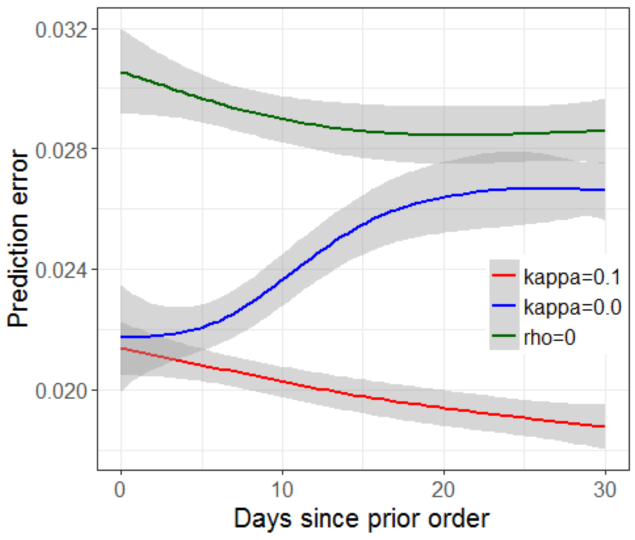
In Figure 4.3(a) we show box plots of dimensional average of mean squared error over the 206K customers' predictions as a function of κ. As mentioned, when $\kappa = 0$, the MM-RNN reduces to its base LSTM model. An increase in κ indicates that this RNN prediction is being forgotten more quickly as the time between orders increases and the customer-level base distribution is being used. We see that performance improves as κ increases, followed by a decrease in performance. Clearly for this data a combination of sequential/non-sequential modeling that takes into consideration customer-level effects and the time between orders is appropriate.

In Figure 4.3(b), we break down these results for $\kappa \in \{0, 0.1\}$ using the output of the run closest to the mean of their corresponding box plots in Figure 4.3(a). We also show results for $\rho \equiv 0$, which reduces the MM-RNN to an exchangeable, i.i.d. model conditioned on ϕ_d for customer d. Here, we show the mean and standard deviation of the prediction errors as a function of days between the previous order and the predicted order.

As we expected, the RNN ($\kappa = 0$) makes worse predictions as this time lag increases, likely because it relies completely on previous sequential information that is less useful in this case. The MM-RNN ($\kappa = 0.1$) is able to adapt and focus more on using the base distribution defined by ϕ_d for customer d. In fact, the performance slightly improves, perhaps indicating that as more time passes the customer runs out of more things and makes and order based on a non-sequential



(a)



(b)

Figure 4.3: Exploratory results on Instacart grocery dataset. (a) Box plots of MSE as a function of κ over multiple runs for $t_0 = 1$. When $\kappa = 0$, the MM-RNN reduces to its base LSTM model. An increase in κ indicates that this RNN prediction is being forgotten more quickly as the time between orders increases and the customer-level base distribution is being used instead. As is evident, a combination of sequential/non-sequential modeling gives more accurate predictions. (b) The error between the aisle distribution and predicted distribution for the last order as a function of time passed since the previous order. We use our basic MM-RNN model (with $\kappa = 0.1$) and compare with an LSTM RNN (equivalent to $\kappa = 0$). As is evident, the LSTM decreases in predictive performance as more time passes between observations. When $\rho \equiv 0$, the model reduces to an exchangeable model on orders, giving further support to our belief in the decreasing sequential value as time lag increases.

distribution on aisles representing that customer's overall preference. In other words, guessing precisely what a customer needs next is inherently more difficult than guessing what that customer needs "when the cupboard is empty." The RNN does not adapt well here, while our simple modification does. We also observe that when the time lag decreases our model still outperforms the RNN. This may be due to the fact that the learned RNN in the MM-RNN was able to better focus on the meaningful sequential content in the data during inference, while the vanilla RNN considers all parts of the sequence as equally meaningful.

Significantly, when $\rho = 0$ we see the same MM-RNN pattern, only worse since no sequential information is being modeled. As time lag increases, the observations from a customer are more approximately conditionally i.i.d., while when the time lag decreases sequential information is important when considering what does and doesn't need to be purchased. This shows that our approach can meaningfully adapt by blending sequential and non-sequential information in the data.

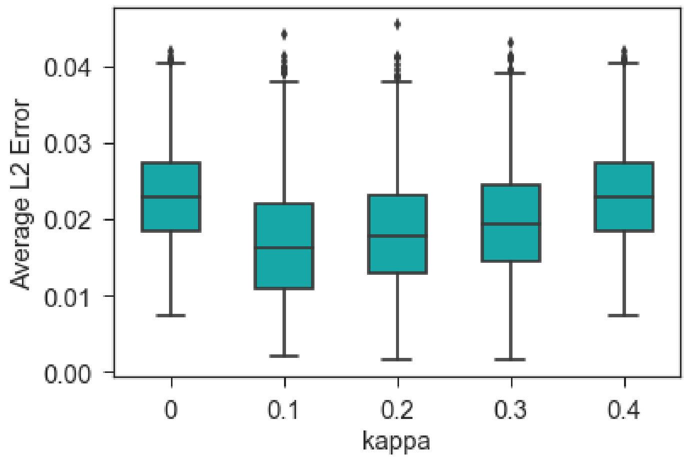
4.4.1.2 Product level model

We also experiment at the product level using the MM-RNN topic model discussed in Section 4.3.2. To initialize the non-negative topic matrix B, we run stochastic LDA [62] one the individual orders as documents to learn 25 topics and use the means of their respective q distributions as initialization. We use the products as vocabulary, but we aggregate products that were purchased less than 20 total times by their aisle. As a result, B is an approximately $36K \times 25$ matrix with topics on the columns. When we ran the MM-RNN model, we then updated B using the multiplicative update rule of [78].

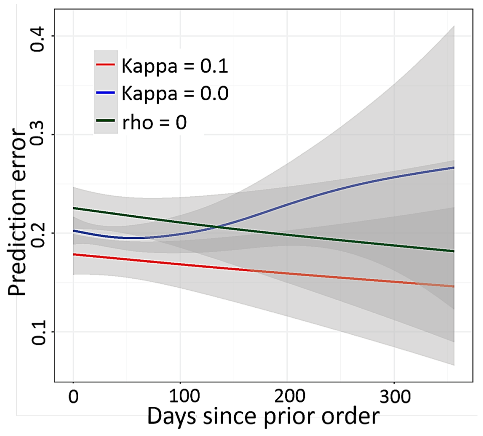
Figure 4.4(a) shows the box plots of 50 experiments with random initializations for multiple values of κ and $t_0 = 1$. These values are normalized to be the mean squared error averaged over the 36K dimensions of all 206K predictions. The conclusions for this MM-RNN approach to the dynamic topic model is the same as in Section 4.4.1.1; at $\kappa = 0$ the model reduces to an LSTM-RNN. We see a clear improvement as κ increases, followed by a decline.

4.4.2 UK online retail dataset

In this section, we present results on the UK online retail dataset from the UCI repository.The data contains information about all transactions in 2010 and 2011 from UK-based and registered non-store online retailers. Similar to the Instacart dataset, this data contains the timestamp of orders purchased by 4373 users from 4070 different products and their corresponding quantity, giving a total of 500K orders. However, in this case the time lag between orders is not truncated



(a)



(b)

Figure 4.4: (a) Boxplot of MSE using the aisle level data when $t_0 = 1$ for Instacart. Similar to the product level case, a combination of sequential/nonsequential modeling gives more accurate predictions. (b) Exploratory results on UK retail dataset. Prediction error for the last order of the users as a function of days since their prior purchase. The details of what is being shown is the same as Figure 4.3(b) (please see caption for description), only cross entropy was used here, and the time lag is not truncated at 30 days in the dataset. The wider error bars are due to the fewer customers and greater variation in time lag.

at 30 days. For each user, we aggregated the daily purchases, and then normalized them to produce vectors of probabilities.

In Figure 4.4(b) we again plot the prediction error using cross entropy loss as a function of time lag between the two final orders for each customer. Since the lag is not truncated to 30 days we are able to plot up to 365 days, but note that there are far fewer samples meaning that the error bars are much wider. Here we observe the same meaningful pattern as in Figure 4.3(b). We note that at around 150 days the RNN (blue) actually performs worse than the non-temporal i.i.d. model (green). MM-RNN (red) outperforms both, again indicating that better RNN and user-specific i.i.d. components were able to be learned by accounting for time lag.

Table 4.1: Quantitative evaluation comparing with other approaches using mean squared error (MSE) and cross entorpy loss (CEL). Direct imputation approaches perform the worst. LSTM-based approaches tend to outperform GRU-based approaches, while our time-adaptive approach tends to improve performance of these respective architectures. Thus, the proposed MM-RNN approach is able to use time between observations to improve predictive performance.

Dataset	Instacart (aisle)		Instacart (product)		UK Retail	
Loss	MSE	CEL	MSE	CEL	MSE	CEL
Impute Mean	0.0437	0.225	0.0877	0.295	0.0593	0.272
Impute Forward	0.0402	0.269	0.0673	0.283	0.0547	0.275
Impute Zero	0.0611	0.318	0.0898	0.349	0.0602	0.304
Che, et al. [21] (GRU-based)	0.0283	0.264	0.0681	0.201	0.0311	0.185
LLSTM [138]	0.0207	0.219	0.0173	0.183	0.0264	**0.168**
LSTM RNN (vanilla)	0.0297	0.288	0.0229	0.261	0.0302	0.193
MM-RNN with LSTM	**0.0192**	**0.214**	**0.0153**	**0.181**	**0.0225**	0.174
MM-RNN with GRU	0.0216	0.223	0.0262	0.205	0.0294	0.213

4.4.3 Quantitative evaluation with other approaches

For all these experiments we compare with various imputation strategies described in [82, 103]. We call these three techniques: 1) Impute Mean, which fills in any missing time step with the global mean; 2) Impute Forward, which fills in missing time points with a copy of the most recent observation; 3) Impute Zero, which fills in missing time points with a vector of zeros. We also compare with [21], a method that also uses a continuous-time weighting strategy to account for different time lags. However, this approach does not take a mixed membership perspective by learning group-level parameters, and the weighting strategy is within the RNN itself, rather than outside of the RNN as in our MM-RNN model. We also compare with [138], which considers an evolving sequence of topics without adjustments for the time interval. In this experiment, we used the Topic LLA model and reported the best result when setting the number of topics to 25, 50 or 100 for each setting. As a special case of our model, we compare with with the vanilla LSTM-RNN ($\kappa = 0$), which represents a purely sequential model with shared parameters. Finally, we compare to the case when we use GRU units instead of LSTM in our model.

We show these results in Table 4.1 with two different choices of loss functions: mean squared error (MSE) and cross entropy loss (CEL) with normalizing input data over the dimensions. As is clear, all imputation methods significantly hurt performance by creating unhelpful sequential information for the RNN that do not help the RNN learning or predictions. While [21] often has better performance relative with the Impute methods, it performs worse than our proposed method since this RNN architecture does not do any group-level modeling, meaning every user's order sequence is treated as being i.i.d.; this indicates the advantage of a mixed membership approach for this type of problem. The MM-RNN also improves over the vanilla RNN with LSTM, which simply ignores the time stamps of the sequences. We can also see that the MM-RNN frequently outperforms [138], which like the vanilla RNN also does not model varying time intervals between the samples. We also see that LSTM performs better than GRU on these data sets for our approach.

4.5 Conclusion

We have presented a mixed membership recurrent neural network (MM-RNN) approach for modeling multiple sequences. The model was motivated by the observation that, in many sequential data sets the sequential information is not of the same value across the sequence. As more time passes between observations, the distribution on the next observation may be better modeled as independent from some initial group-specific distribution. To this end, we made a simple modification to the RNN architecture by generating a unique base vector for each group and use a weighted combination of this base vector with the RNN hidden state to make predictions. The weight emphasizes the RNN in the part of the sequences that is densely sampled, and emphasizes the group-specific i.i.d. model when two consecutive observations are spread far apart in time. We demonstrated on two online shopping data sets that this combination of sequential/non-sequential modeling can allow for the RNN to focus on learning to make better predictions when sequential information is meaningful, and to defer to the base model when much time has passed in a smooth transition.

Chapter 5

Convex Relaxation for Variational Inference

In this chapter, I present a new technique for solving non-convex variational inference optimization problems [43]. Variational inference is a widely used method for posterior approximation in which the inference problem is transformed into an optimization problem. For most models, this optimization is highly non-convex and so hard to solve. I introduce a new approach to solving the variational inference optimization based on convex relaxation and semidefinite programming that further will be extended to other applications in the next chapter. Our theoretical results guarantee very tight relaxation bounds that get nearer to the global optimal solution than traditional coordinate ascent. We evaluate the performance of our approach on regression and sparse coding.

5.1 Introduction

A major challenge of Bayesian modeling is posterior inference. For many models this requires calculating normalizing integrals that neither have a closed form, nor are solvable numerically in polynomial time. There are two fundamental approaches to addressing the posterior inference problem. One uses Markov chain Monte Carlo (MCMC) sampling techniques that are asymptotically exact. However, these methods tend to be slow compared with point-estimates and not scalable to large datasets [46, 58]. Mean-field variational inference is another approach that approximates the posterior distribution by first defining a simpler family of distributions and then finding a member that is closest to the desired posterior [67] according to the KullbackLeibler (KL) divergence. This turns the inference problem into an optimization problem. However, this introduces new challenges due to the resulting non-convex optimization.

In this chapter, I present a method to deal with the non-convexities in variational inference (VI) optimization for conjugate models that achieve near globally optimal solutions. Our method is based on convex relaxation and semidefinite programming (SDP). In our approach, an SDP relaxation converts a non-convex polynomial optimization of vector parameters to a convex optimization with matrix parameters via a lifting technique. We call this approach convex relaxation for variational inference (CRVI). The exactness of the relaxation can then be interpreted as the existence of a low-rank solution to this SDP. Our main contribution is to solve this variational optimization problem in an accurate way and provide theoretical guarantees for the exactness of our solution using graph theoretic tools. To the best of our knowledge, this is the first time that a relaxation for variational inference could guarantee and produce optimal solutions that are either globally optimal solution or very close to it. Our experimental results demonstrate the effectiveness of CRVI compared with coordinate ascent for sparse regression and sparse coding models.

Convex optimization problems are one of the most important areas of optimization theory. They are guaranteed to have global optimal solutions that can be found with a numerical algorithm. On the other hand, there is no such theory for solving generic non-convex problems. Recent advances in the area of convex optimization provide a variety of methods for approaching and solving non-convex optimization problems exactly or approximately [17, 133, 137]. For instance, several works have studied the existence of a low-rank solution to matrix optimizations with linear or nonlinear constraints [40, 86, 98, 101, 122]. We build on the method in [86] to obtain theoretical bounds for the exactness of CRVI.

There are a number of works that have addressed problems with probabilistic inference using convex optimization methods. These works have mostly focused on convex relaxation for maximum entropy and message passing algorithms [56, 94, 113]. In general, they lack control over the exactness of their approximations in that there is no estimate of the closeness of the solution of the relaxed problem to the optimal solution of the original problem.

In this work, we apply convex relaxation techniques to the optimization problem introduced by variational inference with more focus on the cases where the hardness of the problem is due to quadratic or higher order polynomial terms. We first break down the objective function into two parts, one representing the polynomial and non-convex part and one for the rest of the objective function. In this method, we lift the domain of optimization from vectors to matrices, and capture all of non-convexities in the optimization within the transformed problem. As we show, tight relaxation bounds can be achieved to guarantee near-global optimal solution. We also observe that, in models with many parameters this matrix may be prohibitively large. In this case, we still demonstrate how CRVI can be beneficial by relaxing a locally non-convex problem over a subset of variational parameters.

In Section 5.2 we review variational inference and our proposed convex relaxation technique. In Section 5.3 we illustrate our method and discuss theoretical contributions. In Section 5.4, we show experimental results.

5.2 Background

5.2.1 Variational Inference

Variational inference approximates the posterior distribution of variables in a probabilistic model. Let $\mathcal{D}$ be a dataset that is analyzed with a model having variables in the set θ. The model assumption is $\mathcal{D}|\theta \sim p(\mathcal{D}|\theta)$, $\theta \sim p(\theta)$.

The goal is to calculate the posterior distribution $p(\theta|\mathcal{D})$ after observing the data. Due to complexities in most models, finding the true posterior distribution is a difficult task. Instead, we can approximate it by $q(\theta)$ such that this approximation is close to the true distribution according to some notion of similarity. For variational inference, this closeness is measured by the Kulback-Leibler (KL) divergence. To optimize the KL-divergence, one can observe that

$$\ln p(\mathcal{D}) = \underbrace{\mathbb{E}_q\Big[\ln \frac{p(\mathcal{D},\theta)}{q(\theta)}\Big]}_{\mathcal{L}(q(\theta))} + \underbrace{\mathbb{E}_q\Big[\ln \frac{q(\theta)}{p(\theta|\mathcal{D})}\Big]}_{\mathrm{KL}(q\|p)}, \tag{5.1}$$

and since the LHS is constant, one can minimize KL by maximizing the variational objective function $\mathcal{L}$ over the parameters of a predefined distribution family $q(\theta)$. To define this family in a way that is amenable to optimization, one often assumes that $q(\theta)$ belongs to a family of distributions that factorizes over the variables in θ. Seeking to find parameters for this distribution, ϕ, results in optimizing the following problem,

$$\max_{\phi} \quad \mathcal{L}(q(\theta)) \quad \text{subject to} \quad \phi \in \text{feasible set}, \tag{5.2}$$

where the feasible set is the intersection of possible regions for all of the constraints on the parameters. For a very large set of models, this optimization is non-convex or combinatorial, and hard to solve. Numerical algorithms are only able to achieve a local maximum, and most of the time there is no evaluation about how close this local optimum is to the global one.[1] In this work, we consider the cases where this optimization is non-convex and NP-hard. While the global optimum for these optimizations might not be achievable, we aim to find a local optimum that is close to the global solution. Better local optima assure us that we obtain lower KL-divergence and a more accurate posterior approximation. Without loss of generality, we convert the problem to minimizing $-\mathcal{L}(q(\theta))$ over the same feasible set to make the problem more compatible with the convex optimization framework and notations.

We propose a new optimization approach to VI that we call convex relaxation for variational inference (CRVI). This technique approximates the optimization problem to overcome the issues related to non-convexities. As we will show, CRVI can result in near-global optimal solutions that are not only a better local optima compared to the standard coordinate ascent approach, but also provides a means for assessing closeness to the global optimum.

[1]We note that by this we do not mean how close $q(\theta)$ is to $p(\theta|\mathcal{D})$, but how close we are to optimizing the chosen $q(\theta)$.

5.2.2 Convex Relaxation

We next present the general technique that we adopt and build on in this work in its abstract representation. We then apply it to two specific variational inference optimization problems. Although there are exceptions, polynomial terms in an objective or constraint tend to add non-convexities and make the optimization intractable to solve. The technique that we use deals with these hard polynomial parts by converting them into near-exact tractable terms.

First we note that any polynomial function or expression can be represented as a quadratic function, possibly by introducing new variables [8]. This conversion is straightforward, and every high order term could be broken down into lower order terms by introducing new parameters and quadratic equality constraints. As a result, without loss of generality, we assume that all of the polynomial terms are quadratic. Let the following be a general polynomial optimization problem,

$$\begin{aligned} &\min_{x\in\mathbb{R}^d} \; f_0(x) \\ &\text{subject to } f_k(x) \leq 0 \;\; \text{for } k = 1, \ldots, K, \end{aligned} \tag{5.3}$$

where $f_k = x^\top A_k x + b_k^\top x + c_k$ for $k = 0, \ldots, K$. Since there are no limitations on the coefficient choices, the terms in (5.3) can represent any polynomial optimization or expression.

If all of the matrices $\{A_0, A_1, ..., A_K\}$ are positive semidefinite, the optimization in (5.3) is convex. Otherwise, it is non-convex, and there is no numerical or analytical procedure that guarantees achieving a global optimum. We use a lifting technique that involves changing the variable space from vectors to matrices [17]. More specifically, define F_k and X_k as follows,

$$F_K = \begin{bmatrix} c_k & \frac{1}{2}b_k^\top \\ \frac{1}{2}b_k & A_k \end{bmatrix}, \quad X = \begin{bmatrix} 1 & x^\top \\ x & xx^\top \end{bmatrix} \tag{5.4}$$

Then the equivalent optimization to (5.3) is

$$\begin{aligned}
\min_{X\in\mathbb{R}^{(d+1)\times(d+1)}} \quad & \mathrm{trace}(F_0X) \\
\text{subject to} \quad & \mathrm{trace}(F_kX) \leq 0 \quad \text{for } k = 1, .., K, \\
& X_{1,1} = 1, \quad X \succeq 0, \\
& \mathrm{rank}(X) = 1.
\end{aligned} \tag{5.5}$$

The entry equal to 1 in matrix X is to ensure that we have a way to represent the terms that are linear with respect to x. It should be pointed out that matrix X is designed such that it replaces $[1\ x^\top]^\top \times [1\ x^\top]$. This transformation requires us to be able to decompose back the solution X of optimization (5.5) to get the vector x after solving it. To assure this, X needs to be positive semidefinite and have rank 1.

All terms in (5.5) are linear with respect to X and consequently convex, except for the last constraint on the rank of the matrix. To avoid this non-convex rank constraint, we can simply drop it. By dropping the rank constraint, we achieve an optimization that is linear in terms of a matrix variable that has to be positive semidefinite. As a result, we obtain a semidefinite program (SDP) relaxation for the optimization in (5.3) [130]. Although SDP methods may not be fast in general, by carefully designing them and avoiding redundancies, they can run in a reasonable amount of time. The following shows the relaxed optimization problem,

$$\begin{aligned}
\min_{X\in\mathbb{R}^{(d+1)\times(d+1)}} \quad & \mathrm{trace}(F_0X) \\
\text{subject to} \quad & \mathrm{trace}(F_kX) \leq 0 \quad \text{for } k = 1, .., K, \\
& X_{1,1} = 1, \quad X \succeq 0.
\end{aligned} \tag{5.6}$$

One of the important steps here is to quantify the exactness of this relaxation. Naturally we seek approximations that result in finding global optimal or near-global optimal solutions. The only constraint that we dropped is that the matrix has to be rank 1. Hence, in this relaxation, the final rank of X carries information on the exactness of this approximation. After solving the relaxed semidefinite program, if the rank of the optimal X is 1, we have found the global optimal solution for the original problem (5.3). Otherwise, we reach an approximate solution to the original problem. It should be noted that the lower the rank of the optimal solution of the relaxed problem, the closer the approximation to the global optimal solution of the original problem. Thus, the closer the rank of the optimal solution gets to 1, the closer we are to the global optimal solution. This rank of the relaxed problem helps us measure the closeness of the approximate solution to the global optimal solution of the original problem.

Fortunately, the rank of the solution of the relaxed problem cannot be arbitrary large, as shown by [86]. In fact, it is upper bounded by a property of a defined graph structure for the original problem which is its *treewidth*. The treewidth of an undirected graph is a number associated with the graph that is mainly used for complexity analysis of graphs. It can be calculated from the minimum size of largest node over all tree-decomposition of the graph or from the size of the largest clique in a chordal completion of the graph. The treewidth mainly parametrizes and describes the sparsity of a graph, meaning that sparser graphs tend to have smaller treewidths. The process is to first construct a graph from the original quadratic optimization problem (5.3), and then calculate an upper bound on the rank of the semidefinite relaxation using the treewidth of the constructed graph.

To build the graph, we need to assign a vertex to every entry of the vector $[1\ x^\top]^\top$ and add edges between vertices whose product appears in the objective function or any of the constraints of the original problem (5.3). All of the constants or non-variable coefficients are neglected in this process. For instance, if cross-term $x_i x_j$ appears somewhere in (5.3), we put an edge between vertices that correspond to entry x_i and x_j. Or if term x_k appears, we add an edge between vertices corresponding to x_k and 1 since $x_k = x_k \times 1$. Hence, every term in the optimization problem can be translated into a graph edge. Interestingly, one interpretation of adding entry '1' in the matrix definition (5.4) is to be able to represent linear terms as an edge here in the construction of the graph. The fewer the number of cross terms in the optimization, the fewer edges and the sparser the graph.

Now with the graph constructed, we can find an upper bound for the rank of the optimal solution of the relaxed problem in (5.6). The rank of the optimal solution to the relaxed problem is less than or equal to one plus the treewidth of its *enriched super-graph*. As a result, the lower the treewidth of the graph of the problem, the better approximation to the global optimal solution. As we show in the examples, no matter how large the dimensionality of the matrix X

in (5.6), the rank of the optimal solution matrix will be smaller than or equal to the calculated upper bound.

Overall, in this relaxation and transformation, all approximations are pulled into the rank of the optimal solution. An important advantage of this is that if the structure of the sparsity graph of a problem is good enough for us to have a low upper bound, we can achieve a strong relaxation that gives a near global optimal solution. To show how we use this in variational inference, we use a simple example model next. We then generalize it to other models.

5.3 Convex Relaxation for Variational Inference

5.3.1 CRVI for Bayesian Linear Regression

We first show the proposed CRVI method on two Bayesian linear regression models in which the posterior distribution is approximated with variational inference. We start with a simple model. Consider the dataset $\mathcal{D} = \{x_i, y_i\}_{i=1}^N$ with $x \in \mathbb{R}^d$ and $y \in \mathbb{R}$, and the model,

$$\begin{aligned} y_i &\sim \text{Normal}(x_i^\top w, \alpha^{-1}), \\ w &\sim \text{Normal}(0, \lambda^{-1} I), \\ \alpha &\sim \text{Gamma}(a_0, b_0). \end{aligned} \tag{5.7}$$

The goal is to find $p(w, \alpha|\mathcal{D})$, the posterior distribution of the model parameters given the input data. Since the true posterior is hard to find, we apply variational inference to approximate it. Let $q(w, \alpha)$ denote the approximate posterior density and define

$$\begin{aligned} q(w, \alpha) &= q(w)q(\alpha) \qquad (5.8) \\ &= \text{Normal}(w|\mu, \Sigma)\text{Gamma}(\alpha|a, b), \end{aligned}$$

where the factorization comes from the mean-field approximation. The variational objective $\mathcal{L}$ for this optimization problem is

$$\begin{aligned} \mathcal{L}(q) =& (a_0 - 1)(\psi(a) - \ln b) - b_0 \frac{a}{b} - \frac{\lambda}{2}(\mu^\top \mu + \text{trace}(\Sigma)) \\ &+ \frac{N}{2}(\psi(a) - \ln b) - \sum_{i=1}^N \frac{1}{2}\frac{a}{b}((y_i - x_i^\top \mu)^2 + x_i^\top \Sigma x_i) \\ &+ a - \ln b + \ln \Gamma(a) + (1 - a)\psi(a) + \frac{1}{2}\ln|\Sigma| + \text{const.} \end{aligned} \tag{5.9}$$

where 'const.' is a constant with respect to the variational parameters of this model, $\{a, b, \mu, \Sigma\}$, which this function should be maximized over. This objective function is non-concave with respect to its parameters and coordinate ascent variational updates—in which the parameters are cycled over and locally optimized holding the others fixed during each iteration—using arbitrary initialization will likely only achieve locally optimal solutions. We will next show how CRVI can significantly improve this result. We consider the variational inference optimization problem that minimizes $-\mathcal{L}$ subject to $a, b > 0, \quad \Sigma \succeq 0$.

Our approach is to use the relaxation technique presented in the previous section on the polynomial part of this optimization that contains all of the non-convexities associated with this optimization problem. Consider the following reformulated optimization problem,

$$\begin{aligned}
\min \quad & \sum_{i=1}^{N} \frac{1}{2}((ey_i^2 - 2x_i^\top u + x_i^\top u\mu^\top x_i) + x_i^\top e\Sigma x_i) \\
& + \frac{\lambda}{2}(\mu^\top \mu + \mathrm{trace}(\Sigma)) + b_0 e \\
& - (a_0 - 1)(\psi(a) + \ln c) - \frac{N}{2}(\psi(a) + \ln c) \\
& - a - \ln c - \ln \Gamma(a) - (1-a)\psi(a) - \frac{1}{2}\ln|\Sigma| \\
\text{subject to} \quad & a, c, e > 0, \quad \Sigma \succeq 0, \quad e = ac, \quad u = e\mu.
\end{aligned} \tag{5.10}$$

This optimization is over the variables a, c, e, μ, u, Σ. Note that we introduced new variables c to replace $\frac{1}{b}$, e to represent ac and u to replace $e \times \mu$. This enables us to reformulate the polynomial part as a quadratic optimization problem. Hence, optimization problems (5.9) and (5.10) are identical. We refer to the first two lines of (5.10) as $f(a, c, e, \mu, u, \Sigma)$ which is in polynomial form and contains all of the non-convexities in this problem, while we refer to the rest as $g(a, c, \Sigma)$, which is non-linear and convex. This is due to convexity of negative ψ function for positive scalars as well as the convexity of the negative log and negative entropies. Therefore, by relaxing the first part, we get a convex relaxation for the optimization problem. In order to perform the relaxation, we need to rewrite $f(a, c, e, \mu, u, \Sigma)$ as a quadratic function of a vector variable. Based on the semidefinite relaxation construction in the previous section, we define the following vector

$$\nu = \left[1 \, a \, c \, e \, \mu^\top \, u^\top \, \Sigma_{1,1} \, \Sigma_{1,2} \cdots \Sigma_{d,d} \right]^\top$$

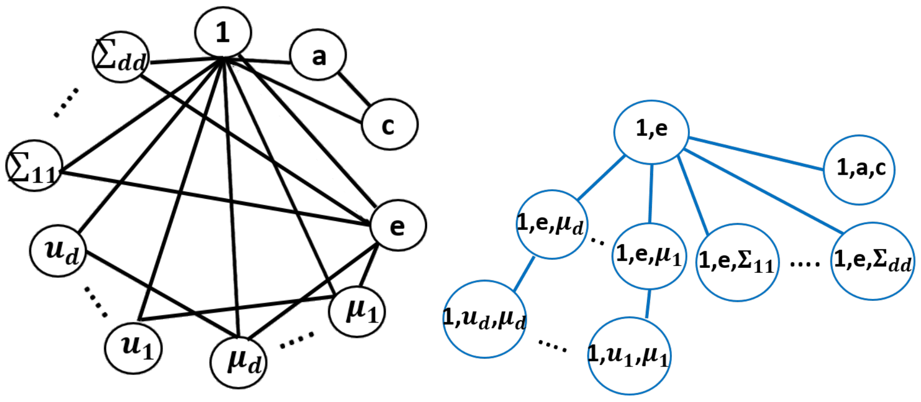


Figure 5.1: Constructed graph for the optimization problem (5.9) on the left side, and its tree decomposition on the right side. Some edges are removed for better legibility of the graphs.

It is easy to see that $f(a, c, e, \mu, u, \Sigma)$ is quadratic with respect to entries of ν. We reformulate the function f to use ν as an argument in f_{CR}. Thus the transformed optimization problem is as follows

$$\begin{aligned} \min_{\nu,a,c,\Sigma} \quad & f_{CR}(\nu) + g(a, c, \Sigma) \\ \text{subject to} \quad & a, c, e \geq 0, \quad e = ac, \quad u = e\mu, \quad \Sigma \succeq 0 \\ & a = \nu_2, \quad c = \nu_3, \\ & \text{vector}(\Sigma) = [\nu_{(5+2*d)} \ldots \nu_{(4+2d+d^2)}] \end{aligned} \tag{5.11}$$

where vector$(\cdot)$ vectorizes the matrix. Convex relaxation can now be defined for the optimization (5.11) by introducing new matrix variable $\mathbf{A} := \nu \times \nu^\top \in \mathbb{S}^{(4+2d+d^2)\times(4+2d+d^2)}$ and following the relaxation steps. $\mathbf{A}$ in this formulation plays the role of X in optimization (5.6). The following proposition gives our theoretical bounds for the exactness of this relaxation.

5.3.1.0.1 Proposition 1. The matrix solution obtained by CRVI for (5.11) has a rank less than or equal to 3.

Proof. Figure 5.1 shows the constructed graph for the original quadratic optimizations (5.9) on the left side, and its tree decomposition on the right side. *Treewidth* is the cardinality of the largest vertex in a graph's tree decomposition minus 1 where its *enriched super-graph* is constructed. Since the cardinality of the largest vertex in its tree decomposition is 3, its treewidth is 2. This guarantees that the rank of the optimal solution of CRVI is upper bounded by 3. □

Note that in Figure (5.1) on the left side, vertex 1 is connected to e, all u entries and all Σ entries. Similarly, e is connected to all entries of μ and Σ. Big blue circles on the right side show the bag of nodes created in the tree decomposition construction.

Although the dimensionality of this optimization can be very large $((4+2d+d^2)\times(4+2d+d^2))$, the rank of its solution is very low (upper bounded by 3 here). This indicates that the relaxation result will be in a close neighborhood of the global optimal solution considering the fact that a rank 1 solution specifies the global optimal solution. Furthermore, this bound exists regardless of dimensionality or scale of the input data.

5.3.2 Model Expansion Using Sparse Priors

We next generalize the Bayesian linear regression model by including dimension specific precisions to w that can be learned to prune irrelevant coefficients in a similar spirit as the Lasso [125]. This model is also known as the relevance vector machine or automatic relevance determination [10]. It modifies the Bayesian linear regression model by defining a separate prior on the diagonal entries of the covariance matrix of w as follows,

$$\begin{aligned} y_i &\sim \text{Normal}(x_i^\top w, \alpha^{-1}), \\ \alpha &\sim \text{Gamma}(a_0, b_0), \\ w &\sim \text{Normal}(0, \text{diag}(\lambda_1, \dots, \lambda_d)^{-1}), \\ \lambda_k &\sim \text{Gamma}(m_0, l_0). \end{aligned} \tag{5.12}$$

Defining a posterior approximating variational distribution q as in the previous case, we now include $q(\lambda_k) =$ Gamma(m_k, l_k) for $k = 1, \dots, d$. Calculating the objective results in the same form as before,

$$\begin{aligned}
&\mathcal{L}(a, b, m_1, \ldots, m_d, l_1, \ldots, l_d, \mu, \Sigma) = \\
&-\sum_{i=1}^{N} \frac{1}{2}\frac{a}{b}((y_i - x_i^\top \mu)^2 + x_i^\top \Sigma x_i) + \frac{N}{2}(\psi(a) - \ln b) \\
&+\sum_{i=1}^{d}(\psi(m_i) - \ln(l_i)) - \frac{1}{2}(\mu^\top \mathrm{diag}(\frac{m_1}{l_1}, ..., \frac{m_d}{l_d})\mu) \\
&-\frac{1}{2}\mathrm{trace}(\mathrm{diag}(\frac{m_1}{l_1}, ..., \frac{m_d}{l_d})\Sigma) \\
&+\sum_{i=1}^{d}(m_0 - 1)(\psi(m_i) - \ln(l_i)) - l_0\frac{m_i}{l_i} \\
&+(a_0 - 1)(\psi(a) - \ln b) - b_0\frac{a}{b} + \frac{1}{2}\ln\ |\Sigma| \\
&+a - \ln b + \ln\Gamma(a) + (1 - a)\psi(a) \\
&+\sum_{i=1}^{d}(m_i - \ln l_i + \ln\Gamma(m_i) + (1 - m_i)\psi(m_i)) + \mathrm{const.}
\end{aligned} \tag{5.13}$$

By reformulating this objective appropriately for convex relaxation, the procedure is very similar to the simpler model. We introduce new variables to replace high order polynomial terms. These new variables are

$$s_i = \frac{1}{l_i},\ r_i = m_i s_i,\ \ \zeta_i = r_i\mu_i \quad \text{for } i = 1, \ldots, d. \tag{5.14}$$

Repeating the relaxation steps described earlier, we achieve a convex relaxation for the optimization of (5.13). Similar to the simpler model, we can achieve the following theoretical result.

5.3.2.0.2 Proposition 2. The matrix solution obtained by CRVI for (5.13) has a rank less than or equal to 3.

The graph structure and tree decomposition for this problem is very similar to the simpler model in (5.3.1), and the same theoretical upper bounds are guaranteed. This strong upper bound exists regardless of the dimensionality of data or size of the input, even though this Bayesian model has a more complex prior structure and many more model parameters. Still, this is only a bound; as we will show in the experiments section the actual rank of the solution to the relaxed optimization is less than 3, and in fact is very close to 1. This means that although the theoretical bound assure us that the rank is less than or equal to 3, in practice on real data sets we can get almost exactly the global optimal solutions of the original problem.

5.3.3 CRVI for Nonparametric Factor Analysis

We illustrate CRVI on a more complex model, Bayesian nonparametric factor analysis [96] of data $\mathcal{D} = \{x_i \in \mathbb{R}^d\}_{i=1}^N$. This will also allow us to propose another modification for the application of this framework due to the much larger number of parameters in the model. The model is

$$\begin{aligned} x_i &\sim \text{Normal}(WZ_iC_i, \sigma^2 I), \\ C_i &\sim \text{Normal}(0, \lambda^{-1} I), \\ \pi_k &\sim \text{Beta}(\alpha\tfrac{\gamma}{K}, \alpha(1-\tfrac{\gamma}{K})), \\ z_{i,k} &\sim \text{Bernoulli}(\pi_k), \\ Z_i &= \text{diag}(z_{i,1}, \dots, z_{i,K}), \end{aligned} \tag{5.15}$$

where $k = 1, ..., K$ are the latent factor indexes. In the limit $K \to \infty$ this converges to a nonparametric beta process model [97]. In addition, due to the model specifications in (5.15), a sparse representation in enforced by beta-Bernoulli prior for Z.

Given a matrix $W \in \mathbb{R}^{d\times K}$, for each vector x_i we seek a sparse zero-one coding Z_i of this vector as well as weight coefficients C_i. The Z's specify which factors in W are used to represent the data, while the C's indicate the weights of those selected factors. In this model we will seek to find the posterior distribution of C as well as point estimates for Z as well as W. Therefore, the algorithm is actually EM and not variational inference since there is no forced factorization of q. However, we do this to focus on another area where CRVI may be useful, as described below.

For each data point i we define $q(C_i)$ =Normal$(C_i|\mu, \Sigma)$. Here, we only focus on learning the local variables for a specific data point x_i, being Z_i, C_i. Therefore, we drop the subscripts below. The optimization problem corresponding to this part of the model is

$$\begin{aligned} \min_{Z,\mu,\Sigma} \quad & \frac{1}{2\sigma^2}(x - WZ\mu)^\top (x - WZ\mu) \\ & + \frac{1}{2\sigma^2}\text{trace}(WZ\Sigma ZW^\top) \\ & + \frac{\lambda}{2}\mu^\top\mu + \frac{\lambda}{2}\text{trace}(\Sigma) - \frac{1}{2}\log(|\Sigma|) + Z^\top h \\ \text{subject to} \quad & Z_{k,k} \in \{0,1\}, \ \text{for } k = 1, ..., K, \quad \Sigma \succeq 0 \end{aligned} \tag{5.16}$$

where h is a constant vector with respect to optimization variables. Note that this optimization can be done in parallel for data points due to their independence. All of objective terms are polynomial with respect to the optimization variables. In addition, the log term is also convex with respect to Σ. To make all of the constraints quadratic, we replace the zero or one constraint for $Z_{k,k}$ with $Z_{k,k}^2 - Z_{k,k} = 0$. Therefore, we obtain a non-convex optimization with polynomial terms containing all of the non-convexities.

5.3.3.0.3 Motivation and discussion. Following the steps described in the previous section, we are able to define the convex relaxation optimization for this problem. Another novelty introduced here is that we have not relaxed the *entire* problem globally, which is computationally impossible for a model of this size (the dimensionality of X would be too massive). Instead, we only relaxed *locally* on the parameters for each observation. However, since optimizing over C and Z is both non-convex and combinatorially hard, we use this model to illustrate a proposed approach to *local relaxation* of the objective. Contrasting this with coordinate ascent, which would update one variable holding another fixed, we anticipate that this can find better local optimal values over *subsets* of parameters, and therefore hopefully over the entire objective function. After constructing the graph of this problem, we find that the rank of the optimal solution of the relaxed problem is upper bounded by 3. Accordingly, we anticipate to find near-global optimal solutions over these interacting local parameters.

5.3.4 CRVI in General Form

Following the ideas introduced by these examples, we present CRVI as a general framework. Let us consider the generic variational inference problem in (5.2). We split the objective into two functions, one containing polynomial terms, f, and one for the remaining parts, g. Transforming f to be a quadratic function, possibly by adding new constraints and variables, we get the optimization

$$\begin{aligned} \min_{\varphi^{(1)},\varphi^{(2)}} \quad & f(\varphi^{(1)}) + g(\varphi^{(2)}) \\ \text{subject to} \quad & \varphi^{(1)},\ \varphi^{(2)} \in \text{feasible set}. \end{aligned} \tag{5.17}$$

Note that $\varphi^{(1)}$ and $\varphi^{(2)}$ might have overlapping parameters. To complete the relaxation, we introduce a new matrix variable $\Phi^{(1)}$ and obtain CRVI for the general form,

$$\begin{aligned} \min_{\Phi^{(1)},\varphi^{2}} \quad & f(\Phi^{(1)}) + g(\varphi^{(2)}) \\ \text{subject to} \quad & \Phi^{(1)},\ \varphi^{(2)} \in \text{feasible set}, \\ & {\Phi^{(1)}}_{1,1} = 1, \quad \Phi^{(1)} \succeq 0. \end{aligned} \tag{5.18}$$

If g is a convex function, (5.18) is a convex optimization problem solvable in polynomial time. By constructing the graph for this relaxation approximation bounds can be achieved. The lower the rank of the optimal solution $\Phi^{(1)}_{\text{opt}}$, the more exact the approximation. As seen in the above examples, variational inference do have this structure, for which low rank recovery and near-global optimal solutions are guaranteed. In the cases where g is non-convex, CRVI could be used to partially convexify the optimization problem. We can reduce the hardness related to f with this relaxation technique, get approximation bounds, and improve the results compared to the cases where we have to deal with both non-convex f and g.

5.4 Experimental Results

5.4.1 CRVI for Sparse Bayesian Linear Regression

We focus on comparing the optimal value of the variational objective calculated by our method CRVI in Section (5.3.2), and using coordinate ascent variational inference (CAVI) which is the standard method for variational optimization. We implemented CRVI code using CVX, which is a package for specifying and solving convex programs [53, 54]. We experiment on 9 datasets from the UCI repository with various sizes and dimensions. These data sets are: Iris, Birth rate and economic growth, Yacht, Pima Indian diabetes, Bike sharing, Parkinson data, Wisconsin breast cancer (WDBC), Online news popularity, Year of release prediction for a million songs. We experimented using 100 different hyper-parameter settings and initial values for each dataset. Table (5.1) shows some details about these datasets, as well as the average running time for our simulations and the average rank of the optimal solution found by CRVI.

Table 5.1: Information about the datasets, running time of the algorithms, and rank of the found solution using CRVI. We see that CRVI is slower than CAVI (coordinate ascent). However, the rank of the found CRVI solution is near 1 (and less than the theoretical upper bound of 3), indicating a solution nearer the global optimum. This is confirmed in Figure 5.2.

DataSet	Dim.	# of Samples	CAVI time (s)	CRVI time (s)	Rank
Birth Rate & Econ	4	30	0.281	1.115	1.11
Iris	4	150	0.231	1.807	1.20
Yacht	6	308	0.402	2.111	1.10
Pima Indian Diabetes	8	768	0.571	3.040	1.67
Bike Sharing	13	731	0.884	6.749	1.61
Parkinson	21	5875	0.962	7.309	1.98
WDBC	31	569	1.059	10.766	1.73
Online News Popularity	58	39644	9.341	15.223	1.52
Year Prediction Songs	90	515345	18.809	22.050	1.78

As can be seen, CRVI is slower than CAVI, which is not unexpected. Although the actual dimensionality of the semidefinite matrix variables for these datasets varies from 28×28 to 8284×8284, the average ranks found show that, regardless of the size of the data, the rank remains small and close to 1. This means that the CRVI is able to find nearly-global optimal solutions, considering that a rank 1 solution gives the exact global optimum solution. To evaluate the improvement according to the variational objective function, for each simulation of each dataset we subtracted the local optimal value of CAVI from CRVI, and divided it by optimal value found by CAVI to get the relative improvement to the maximization problem. We show a summary of these results in a boxplot for each dataset in Figure 5.2. As can be seen, CRVI significantly improved the local optimal solution of the optimization over coordinate ascent, which can be interpreted as finding a more accurate posterior approximation.

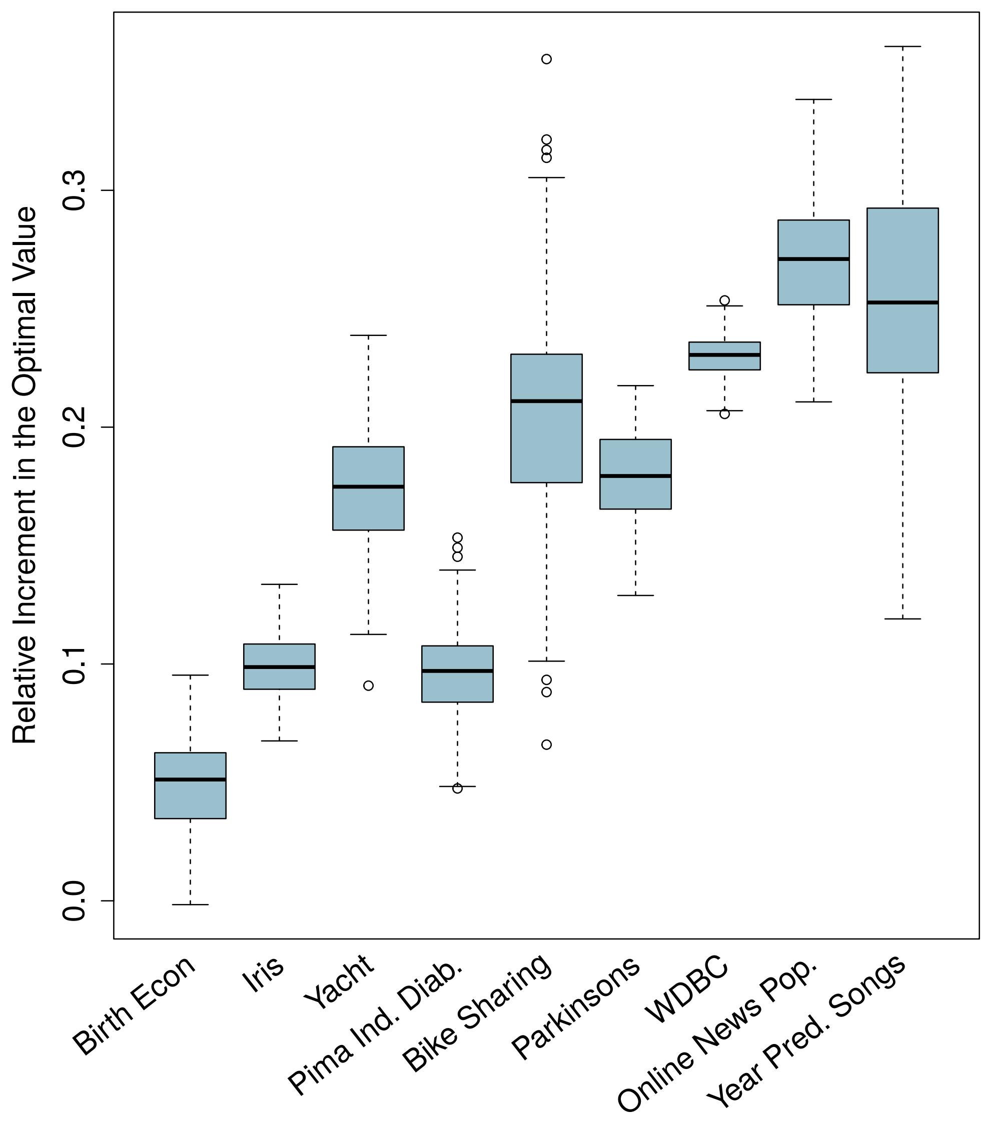


Figure 5.2: Boxplot of relative improvement in the calculated local optimal value of CRVI compared to CAVI. Each box represents the summary of the fractional improvement of CRVI over CAVI for 100 simulations using different prior hyper-parameters and initializations. After calculating the respective local optimal variational objective functions, the value found by CAVI is subtracted from the value from CRVI and divided by the values from CAVI to obtain the relative improvement score. As is evident, CRVI gave a significant improvement over CAVI.

5.4.2 CRVI for Nonparametric Factor Analysis

We also compare the accuracy of CRVI for sparse signal representation for dictionary learning with K-SVD [1] on synthetic data. K-SVD uses orthogonal matching pursuits (OMP) to encode each signal in a dictionary [128], which is also learned during the optimization process. Our goal is to compare the number of correctly recovered entries in the binary Z. We generate $N = 300$ observations of $D = 100$ dimensions and set $K = 100$ and $\lambda = 0.1$. We change the sparsity level of the generated Z over different simulations.

In Figure 5.3, the x-axis represents the probability of a '1' in each entry of Z when generating this binary encoding, while the y-axis shows the percentage of correctly recovered values in Z over the entire data set. As can be seen, CRVI is able to better learn the correct values for Z by finding the correct sparsity. Figure 5.4 shows the percentage of correctly recovered 1's for CRVI and KSVD. As can be seen from the figure, CRVI has a better performance in recovering the correct locations of 1's in the original Z matrix. Also, since we are focusing on the local optimal solution over Z and C as discussed in Section 5.3.3, we use the correct W in this experiment. Therefore, KSVD actually reduces to OMP in this experiment.

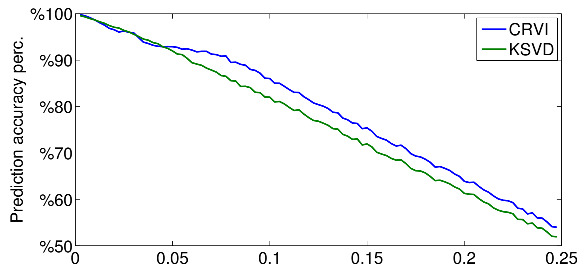


Figure 5.3: The fraction of agreement in the recovered Z's with original Z using CRVI and K-SVD (here, OMP). The x-axis shows the probability of a 1 in every entry of the original sparse matrix.

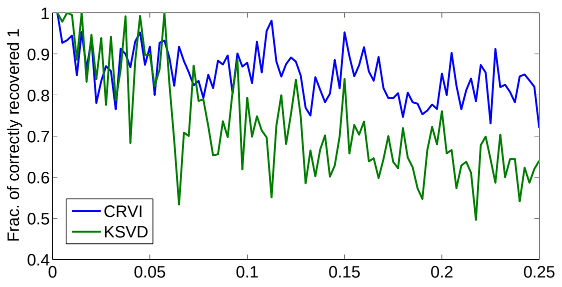


Figure 5.4: The fraction of correctly recovered 1's in the original Z using CRVI and K-SVD (here, OMP). The x-axis shows the probability of a 1 in every entry of original sparse matrix.

5.5 Discussion

Convex relaxations are a powerful technique for approximating (convexifying) hard optimization problems associated with variational inference. However, one of the caveats of this method is its runtime complexity, arising mostly from the positive semidefinite constraint. Fortunately, recent advances in this area have suggested faster ways to impose these types of constraints by breaking them into several smaller-sized semidefinite constraints. This significantly improves the running time of these types of relaxations [68]. We expect that incorporating these techniques can improve the computational performance of this algorithm. Another future direction is to find tighter bounds for the relaxation exactness using the treewidth measure. Finding the exact treewidth of a graph is an NP-hard problem in general, and the bounds given in this work used the treewidth's that were within our computational power. There may be better ways to reach smaller treewidth's and make the theoretical bounds tighter. The observed ranks in Table (5.1), smaller than the theoretical upper bound of 3, indicate that there is room for improvement in the theory in this direction.

5.6 Conclusion

In this chapter, I presented convex relaxation for variational inference (CRVI), a method to learn parameters of approximate posterior distributions using mean-field variational inference. We focused on Bayesian linear regression and sparse coding models. By lifting the domain of the optimization, we were able to relax the non-convex parts of the variational objective function and approximate the variational parameters. Graph theoretic tools enabled us to quantify the exactness of this approximation, and estimate the closeness of the obtained solution to the global optimal solution. We showed that CRVI can significantly improve the traditional coordinate ascent (CAVI) optimization technique on various datasets for sparse Bayesian linear regression and sparse coding for nonparametric factor analysis.

Chapter 6

Convex Relaxation for Distributed Control Problem

This chapter is concerned with the optimal distributed control (ODC) problem for discrete-time deterministic and stochastic systems. The objective is to design a fixed-order distributed controller with a pre-specified structure that is globally optimal with respect to a quadratic cost functional. It is shown that this NP-hard problem has a quadratic formulation, which can be relaxed to a semidefinite program (SDP). If the SDP relaxation has a rank-1 solution, a globally optimal distributed controller can be recovered from this solution. By utilizing the notion of treewidth, it is proved that the nonlinearity of the ODC problem appears in such a sparse way that an SDP relaxation of this problem has a matrix solution with rank at most 3. Since the proposed SDP relaxation is computationally expensive for a large-scale system, a computationally-cheap SDP relaxation is also developed with the property that its objective function indirectly penalizes the rank of the SDP solution. Various techniques are proposed to approximate a low-rank SDP solution with a rank-1 matrix, leading to recovering a near-global controller together with a bound on its optimality degree. The above results are developed for both finite-horizon and infinite horizon ODC problems. While the finite-horizon ODC is investigated using a time-domain formulation, the infinite-horizon ODC problem for both deterministic and stochastic systems is studied via a Lyapunov formulation. The SDP relaxations developed in this work are exact for the design of a centralized controller, hence serving as an alternative for solving Riccati equations. The efficacy of the proposed SDP relaxations is elucidated in numerical examples.

6.1 Introduction

The area of decentralized control is created to address the challenges arising in the control of real-world systems with many interconnected subsystems. The objective is to design a structurally constrained controller—a set of partially interacting local controllers—with the aim of reducing the computation or communication complexity of the overall controller. The local controllers of a decentralized controller may not be allowed to exchange information. The term *distributed control* is often used in lieu of decentralized control in the case where there is some information exchange between the local controllers (possibly distributed over a geographical area). It has been long known that the design of a globally optimal decentralized (distributed) controller is a daunting task because it amounts to an NP-hard optimization problem in general [129, 135]. Great effort has been devoted to investigating this highly complex problem for special types of systems, including spatially distributed systems [5, 27, 33, 73, 90], dynamically decoupled systems [15, 69], weakly coupled systems [117], and strongly connected systems [74]. Another special case that has received considerable attention is the design of an optimal static distributed controller [37, 81]. Early approaches for the optimal decentralized control problem were based on parameterization techniques [28, 47], which were then evolved into matrix optimization methods [112, 139]. In fact, with a structural assumption on the exchange of information between subsystems, the performance offered by linear static controllers may be far less than the optimal performance achievable using a nonlinear time-varying controller [135].

Due to the recent advances in the area of convex optimization, the focus of the existing research efforts has shifted from deriving a closed-form solution for the above control synthesis problem to finding a convex formulation of the problem that can be efficiently solved numerically [6, 29, 34, 89, 104]. This has been carried out in the seminal work [109] by deriving a sufficient condition named quadratic invariance, which has been specialized in [116] by deploying the concept of partially order sets. These conditions have been further investigated in several other papers [72, 79, 110]. A different approach is taken in the recent papers [123] and [106], where it has been shown that the distributed control problem can be cast as a convex optimization for positive systems.

There is no surprise that the decentralized control problem is computationally hard to solve. This is a consequence of the fact that several classes of optimization problems, including polynomial optimization and quadratically-constrained quadratic program as a special case, are NP-hard in the worst case. Due to the complexity of such problems, various convex relaxation methods based on linear matrix inequality (LMI), semidefinite programming (SDP), and second-order cone programming (SOCP) have gained popularity [18, 131]. These techniques enlarge the possibly non-convex feasible set into a convex set characterizable via convex functions, and then provide the exact or a lower bound on the optimal objective value. The effectiveness of these techniques has been reported in several papers. For instance, [50] shows how SDP relaxation can be used to find better approximations for maximum cut (MAX CUT) and maximum 2-satisfiability (MAX 2SAT) problems. Another approach is proposed in [51] to solve the max-3-cut problem via a complex SDP. The approaches in [50] and [51] have been generalized in several papers, including [59, 93].

Semidefinite programming relaxation usually converts an optimization with a vector variable to a convex optimization with a matrix variable, via a lifting technique. The exactness of the relaxation can then be interpreted as the existence of a low-rank (e.g., rank-1) solution for SDP relaxation. Several papers have studied the existence of a low-rank solution to matrix optimizations with linear or nonlinear (e.g., LMI) constraints. For instance, the papers [85, 101] provide upper bounds on the lowest rank among all solutions of a feasible LMI problem. A rank-1 matrix decomposition technique is developed in [122] to find a rank-1 solution whenever the number of constraints is small. We have shown in [76] and [118] that SDP relaxation is able to solve a large class of non-convex energy-related optimization problems performed over power networks. We related the success of the relaxation to the hidden structure of those optimizations induced by the physics of a power grid. Inspired by this positive result, we developed the notion of "nonlinear optimization over graph" in [119–121]. Our technique maps the structure of an abstract nonlinear optimization into a graph from which the exactness of SDP relaxation may be concluded. By adopting the graph technique developed in [121], the objective of the present work is to study the potential of SDP relaxation for the optimal distributed control problem.

In this work, we cast the optimal distributed control (ODC) problem as a non-convex optimization problem with only quadratic scalar and matrix constraints, from which an SDP relaxation can be obtained. The goal is to show that this relaxation has a low-rank solution whose rank depends on the topology of the controller to be designed. In particular, we prove that the design of a static distributed controller with a pre-specified structure amounts to a sparse SDP relaxation with a solution of rank at most 3. This positive result is a consequence of the fact that the sparsity graph associated with the underlying optimization problem has a small treewidth. The notion of "treewidth" used in this work could potentially help to understand how much approximation is needed to make the ODC problem tractable. This is due to a recent result stating that a rank-constrained optimization problem has an almost equivalent convex formulation whose size depends on the treewidth of a certain graph [9]. In this work, we also discuss how to round the rank-3 SDP matrix to a rank-1 matrix in order to design a near-global controller.

The results of this work hold true for both a time-domain formulation corresponding to a finite-horizon control problem and a Lyapunov-domain formulation associated with an infinite-horizon deterministic/stochastic control problem. We first investigate the ODC problem for the deterministic systems and then the ODC problem for stochastic systems. Our approach rests on formulating each of these problems as a rank-constrained optimization from which an SDP relaxation can be derived. With no loss of generality, this work focuses on the design of a static controller. Since the proposed relaxations with guaranteed low-rank solutions are computationally expensive, we also design computationally-cheap SDP relaxations for numerical purposes. Afterwards, we develop some heuristic methods to recover a near-optimal controller from a low-rank SDP solution. Note that the computationally-cheap SDP relaxations associated with the infinite-horizon ODC are exact in both deterministic and stochastic cases for the classical (centralized) LQR and H_2 problems. Although the focus of this chapter is static controllers, its results can be naturally generalized to the dynamic case as well.

We conduct case studies on a mass-spring system and 100 random systems to elucidate the efficacy of the proposed relaxations. In particular, the design of many near-optimal structured controllers with global optimality degrees above 99% will be demonstrated.

This work is organized as follows. The problem is introduced in Section 6.2, and then the SDP relaxation of a quadratically-constrained quadratic program (QCQP) is studied via a graph-theoretic approach. Three different SDP relaxations of the finite-horizon deterministic ODC problem are presented for the static controller design in Section 6.3. The infinite-horizon deterministic ODC problem is studied in Section 6.4. The results are generalized to an infinite-horizon stochastic ODC problem in Section 6.5, followed by a brief discussion on dynamic controllers in Section 6.6. Various experiments and simulations are provided in Section 6.7. Concluding remarks are drawn in Section 6.8.

6.1.1 Notations

$\mathbb{R}$, $\mathbb{S}_n$ and $\mathbb{S}_n^+$ denote the sets of real numbers, $n \times n$ symmetric matrices and $n \times n$ positive semidefinite matrices, respectively. The m by n rectangular identity matrix whose (i, j) entry is equal to the Kronecker delta δ_{ij} is denoted by $I_{m\times n}$ or alternatively I_n when $m = n$. rank$\{W\}$ and trace$\{W\}$ denote the rank and trace of a matrix W. The notation $W \succeq 0$ means that W is symmetric and positive semidefinite. Given a matrix W, its (l, m) entry is denoted as W_{lm}. Given a block matrix $\mathbf{W}$, its (l, m) block is shown as $\mathbf{W}_{lm}$. Given a matrix M, its Moore Penrose pseudoinverse is denoted as M^+. The superscript $(\cdot)^{\text{opt}}$ is used to show a *globally optimal value* of an optimization parameter. The symbols $(\cdot)^T$ and $\|\cdot\|$ denote the transpose and 2-norm operators, respectively. The symbols $\langle \cdot, \cdot \rangle$ and $\|\cdot\|_F$ denote the Frobinous inner product and norm of matrices, respectively. The notation $|.|$ shows the size of a vector, the cardinality of a set or the number of vertices a graph, depending on the context. The expected value of a random variable x is shown as $\mathcal{E}\{x\}$. The submatirx of M formed by rows form the set $\mathcal{S}_1$ and columns from the set $\mathcal{S}_2$ is denoted by $M\{\mathcal{S}_1, \mathcal{S}_2\}$. The notation $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ implies that $\mathcal{G}$ is a graph with the vertex set $\mathcal{V}$ and the edge set $\mathcal{E}$.

6.2 Preliminaries

In this work, the Optimal Distributed Control (ODC) problem is studied based on the following steps:

- First, the problem is cast as a non-convex optimization problem with only quadratic scalar and/or matrix constraints.
- Second, the resulting non-convex problem is formulated as a rank-constrained optimization.
- Third, a convex relaxation of the problem is derived by dropping the non-convex rank constraint.
- Last, the rank of the minimum-rank solution of the SDP relaxation is analyzed.

Since there is no unique SDP relaxation for the ODC problem, a major part of this work is devoted to designing a sparse quadratic formulation of the ODC problem with a guaranteed low-rank SDP solution. To achieve this goal, a graph is associated to each SDP, which is then sparsified to contrive a problem with a low-rank solution.

6.2.1 Problem Formulation

The following variations of the Optimal Distributed Control (ODC) problem are studied in this work.

6.2.1.1 Finite-horizon deterministic ODC problem

Consider the discrete-time system

$$x[\tau+1] = Ax[\tau] + Bu[\tau], \qquad \tau = 0, 1, \ldots, p-1 \tag{6.1a}$$

$$y[\tau] = Cx[\tau], \qquad \tau = 0, 1, \ldots, p \tag{6.1b}$$

with the known matrices $A \in \mathbb{R}^{n\times n}$, $B \in \mathbb{R}^{n\times m}$, $C \in \mathbb{R}^{r\times n}$, and $x[0] = x_0 \in \mathbb{R}^n$, where p is the terminal time. The goal is to design a distributed static controller $u[\tau] = Ky[\tau]$ minimizing a quadratic cost function under the constraint that the controller gain K must belong to a given linear subspace $\mathcal{K} \subseteq \mathbb{R}^{m\times r}$. The set $\mathcal{K}$ captures the sparsity structure of the unknown constrained controller and, more specifically, it contains all $m \times r$ real-valued matrices with forced zeros in certain entries. The cost function

$$\sum_{\tau=0}^{p} \left(x[\tau]^T Q x[\tau] + u[\tau]^T R u[\tau] \right) + \alpha \|K\|_F^2 \tag{6.2}$$

is considered in this work, where α is a nonnegative scalar, and Q and R are positive-semidefinite matrices. This problem will be studied in Section 6.3.

Remark 6.1. *The third term in the objective function of the ODC problem is a soft penalty term aimed at avoiding a high-gain controller. Instead of this soft penalty, we could impose a hard constraint $\|K\|_F \leq \beta$, for a given number β. The method to be developed later can be adopted for the modified case.*

6.2.1.2 Infinite-horizon deterministic ODC problem

The infinite-horizon ODC problem corresponds to the case $p = +\infty$ subject to the additional constraint that the controller must be stabilizing. This problem will be studied through a Lyapunov domain formulation in Section 6.4.

6.2.1.3 Infinite-horizon stochastic ODC problem

Consider the discrete-time stochastic system

$$x[\tau+1] = Ax[\tau] + Bu[\tau] + Ed[\tau], \qquad \tau = 0, 1, \ldots \tag{6.3a}$$

$$y[\tau] = Cx[\tau] + Fv[\tau], \qquad \tau = 0, 1, \ldots \tag{6.3b}$$

with the known matrices A, B, C, E, and F, where $d[\tau]$ and $v[\tau]$ denote the input disturbance and measurement noise, which are assumed to be zero-mean white-noise random processes. The ODC problem for the above system will be investigated in Section 6.5.

The extension of the above results to the design of dynamic controllers will be briefly discussed in Section 6.6.

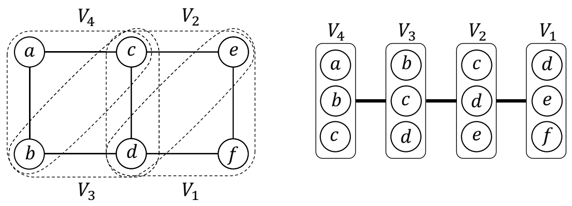


Figure 6.1: A minimal tree decomposition for a ladder graph.

6.2.2 Graph Theory Preliminaries

Definition 6.1. *For two simple graphs $\mathcal{G}_1 = (\mathcal{V}, \mathcal{E}_1)$ and $\mathcal{G}_2 = (\mathcal{V}, \mathcal{E}_2)$ with the same set of vertices, their union is defined as $\mathcal{G}_1 \cup \mathcal{G}_2 = (\mathcal{V}, \mathcal{E}_1 \cup \mathcal{E}_2)$.*

Definition 6.2. *The representative graph of an $n \times n$ symmetric matrix W, denoted by $\mathcal{G}(W)$, is a simple graph with n vertices whose edges are specified by the locations of the nonzero off-diagonal entries of W. In other words, two disparate vertices i and j are connected if W_{ij} is nonzero.*

Consider a graph $\mathcal{G}$ identified by a set of "vertices" and a set of edges. This graph may have cycles in which case it cannot be a tree. Using the notion to be explained below, we can map $\mathcal{G}$ into a tree $\mathcal{T}$ identified by a set of "nodes" and a set of edges where each node of $\mathcal{T}$ contains a group of vertices of $\mathcal{G}$.

Definition 6.3 (Treewidth)**.** *Given a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, a tree $\mathcal{T}$ is called* a tree decomposition of $\mathcal{G}$ *if it satisfies the following properties:*

1. *Every node of $\mathcal{T}$ corresponds to and is identified by a subset of $\mathcal{V}$.*

2. *Every vertex of $\mathcal{G}$ is a member of at least one node of $\mathcal{T}$.*

3. *For every edge (i, j) of $\mathcal{G}$, there should be a node in $\mathcal{T}$ containing vertices i and j simultaneously.*

4. *Given an arbitrary vertex k of $\mathcal{G}$, the subgraph induced by all nodes of $\mathcal{T}$ containing vertex k must be connected (more precisely, a tree).*

Each node of $\mathcal{T}$ is a bag (collection) of vertices of $\mathcal{G}$ and hence it is referred to as bag. The width of $\mathcal{T}$ is the cardinality of its biggest bag minus one. The treewidth of $\mathcal{G}$ is the minimum width over all possible tree decompositions of $\mathcal{G}$ and is denoted by $\mathrm{tw}(\mathcal{G})$*.*

Every graph has a trivial tree decomposition with one single bag consisting of all its vertices. Figure 6.1 shows a graph $\mathcal{G}$ with 6 vertices named a, b, c, d, e, f, together with its minimal tree decomposition $\mathcal{T}$. Every node of $\mathcal{T}$ is a set containing three members of $\mathcal{V}$. The width of this decomposition is therefore equal to 2. Observe that the edges of the tree decomposition are chosen in such a way that every subgraph induced by all bags containing each member of $\mathcal{V}$ is a tree (as required by Property 4 stated before).

Note that if $\mathcal{G}$ is a tree itself, it has a minimal tree decomposition $\mathcal{T}$ such that: each bag corresponds to two connected vertices of $\mathcal{G}$ and every two adjacent bags in $\mathcal{T}$ share a vertex in common. Therefore, the treewidth of a tree is equal to 1. The reader is referred to [14] for a comprehensive literature review on treewidth.

6.2.3 SDP Relaxation

The objective of this subsection is to study SDP relaxation of a quadratically-constrained quadratic program (QCQP) using a graph-theoretic approach. Consider the standard nonconvex QCQP problem

$$\underset{x\in\mathbb{R}^n}{\text{minimize}} \quad f_0(x) \tag{6.4a}$$

$$\text{subject to} \quad f_k(x) \leq 0, \qquad k = 1, \ldots, q, \tag{6.4b}$$

where $f_k(x) = x^T A_k x + 2b_k^T x + c_k$ for $k = 0, \ldots, q$. Define

$$F_k \triangleq \begin{bmatrix} c_k \; b_k^T \\ \\ b_k \; A_k \end{bmatrix}. \tag{6.5}$$

Each f_k has the linear representation $f_k(x) = \langle F_k, W\rangle$ for the following choice of W:

$$W \triangleq [x_0 \quad x^T]^T [x_0 \quad x^T] \tag{6.6}$$

where x_0 is considered as 1. On the other hand, an arbitrary matrix $W \in \mathbb{S}_{n+1}$ can be factorized as (6.6) if and only if it satisfies three properties: $W_{11} = 1$, $W \succeq 0$, and $\text{rank}\{W\} = 1$. In this representation of QCQP, the rank constraint carries all the nonconvexity. Neglecting this constraint yields the convex problem

$$\begin{aligned}
&\underset{W\in\mathbb{S}_{n+1}}{\text{minimize}} && \langle F_0, W\rangle && && (6.7\text{a})\\
&\text{subject to} && \langle F_k, W\rangle \leq 0 && k=1,\ldots,q, && (6.7\text{b})\\
& && W_{11}=1, && && (6.7\text{c})\\
& && W\succeq 0, && && (6.7\text{d})
\end{aligned}$$

known as a semidefinite programming (SDP) relaxation of the QCQP (6.4). The existence of a rank-1 solution for an SDP relaxation guarantees the equivalence of the original QCQP and its relaxed problem.

6.2.4 Connection Between Rank and Sparsity

To explore the rank of the minimum-rank solution of SDP relaxation, define $\mathcal{G}=\mathcal{G}(F_0)\cup\cdots\cup\mathcal{G}(F_q)$ as the **sparsity graph** associated with the problem (6.7). The graph $\mathcal{G}$ describes the zero-nonzero pattern of the matrices $F_0,\ldots,F_q$, or alternatively captures the sparsity level of the QCQP problem (6.4). Let $\mathcal{T}=(\mathcal{V}_\mathcal{T},\mathcal{E}_\mathcal{T})$ be a tree decomposition of $\mathcal{G}$. Denote its width as t and its bags as $\mathcal{B}_1,\mathcal{B}_2,...,\mathcal{B}_{|\mathcal{T}|}$. It is known that given such a decomposition, every solution $W^{\text{ref}}\in\mathbb{S}_{n+1}$ of the SDP problem (6.7) can be transformed into a solution W^{opt} whose rank is upper bounded by $t+1$ [85]. To perform this transformation, a suitable polynomial-time recursive algorithm will be proposed below.

Rank reduction algorithm:

1. Set $\mathcal{T}':=\mathcal{T}$ and $W:=W^{\text{ref}}$.
2. If $\mathcal{T}'$ has a single node, then consider W^{opt} as W and terminate; otherwise continue to the next step.
3. Choose a pair of bags $\mathcal{B}_i,\mathcal{B}_j$ of $\mathcal{T}'$ such that $\mathcal{B}_i$ is a leaf of $\mathcal{T}'$ and $\mathcal{B}_j$ is its unique neighbor.

4. Using the notation $W\{\cdot,\cdot\}$ introduced in Section 6.1.1, define

$$O \triangleq W\{\mathcal{B}_i \cap \mathcal{B}_j, \mathcal{B}_i \cap \mathcal{B}_j\} \tag{6.8a}$$

$$V_i \triangleq W\{\mathcal{B}_i \setminus \mathcal{B}_j, \mathcal{B}_i \cap \mathcal{B}_j\} \tag{6.8b}$$

$$V_j \triangleq W\{\mathcal{B}_j \setminus \mathcal{B}_i, \mathcal{B}_i \cap \mathcal{B}_j\} \tag{6.8c}$$

$$H_i \triangleq W\{\mathcal{B}_i \setminus \mathcal{B}_j, \mathcal{B}_i \setminus \mathcal{B}_j\} \in \mathbb{R}^{n_i \times n_i} \tag{6.8d}$$

$$H_j \triangleq W\{\mathcal{B}_j \setminus \mathcal{B}_i, \mathcal{B}_j \setminus \mathcal{B}_i\} \in \mathbb{R}^{n_j \times n_j} \tag{6.8e}$$

$$S_i \triangleq H_i - V_i O^+ V_i^T = Q_i \Lambda_i Q_i^T \tag{6.8f}$$

$$S_j \triangleq H_j - V_j O^+ V_j^T = Q_j \Lambda_j Q_j^T \tag{6.8g}$$

where $Q_i\Lambda_i Q_i^T$ and $Q_j\Lambda_j Q_j^T$ denote the eigenvalue decompositions of S_i and S_j with the diagonals of Λ_i and Λ_j arranged in descending order. Then, update a part of W as follows:

$$\begin{aligned} W\{\mathcal{B}_j \setminus \mathcal{B}_i, \mathcal{B}_i \setminus \mathcal{B}_j\} := {} & V_j O^+ V_i^T \\ & + Q_j \sqrt{\Lambda_j}\; I_{n_j \times n_i} \sqrt{\Lambda_i}\, Q_i^T \end{aligned}$$

and update $W\{\mathcal{B}_i \setminus \mathcal{B}_j, \mathcal{B}_j \setminus \mathcal{B}_i\}$ accordingly to preserve the Hermitian property of W.

5. Update $\mathcal{T}'$ by merging $\mathcal{B}_i$ into $\mathcal{B}_j$, i.e., replace $\mathcal{B}_j$ with $\mathcal{B}_i \cup \mathcal{B}_j$ and then remove $\mathcal{B}_i$ from $\mathcal{T}'$.

6. Go back to step 2.

Theorem 6.1. *The output of the rank reduction algorithm, denoted as W^{opt}, is a solution of the SDP problem* (6.7) *whose rank is smaller than or equal to $t+1$.*

Proof 6.1. *Consider one run of Step 4 of the rank reduction algorithm. Our first objective is to show that $W\{\mathcal{B}_i \cup \mathcal{B}_j, \mathcal{B}_i \cup \mathcal{B}_j\}$ is a positive semidefinite matrix whose rank is upper bounded by the maximum ranks of $W\{\mathcal{B}_i, \mathcal{B}_i\}$ and $W\{\mathcal{B}_j, \mathcal{B}_j\}$. To this end, one can write:*

$$W\{\mathcal{B}_i \cup \mathcal{B}_j, \mathcal{B}_i \cup \mathcal{B}_j\} = \begin{bmatrix} O & V_i^T & V_j^T \\ V_i & H_i & Z^T \\ V_j & Z & H_j \end{bmatrix} \tag{6.9}$$

where $Z \triangleq W\{\mathcal{B}_j \setminus \mathcal{B}_i, \mathcal{B}_i \setminus \mathcal{B}_j\}$. *Now, define*

$$\begin{aligned} S &\triangleq \begin{bmatrix} H_i \; Z^T \\ \\ Z \;\; H_j \end{bmatrix} - \begin{bmatrix} V_i \\ \\ V_j \end{bmatrix} O^+ \begin{bmatrix} V_i^T \, V_j^T \end{bmatrix} \\ &= \begin{bmatrix} Q_i & 0 \\ \\ o & Q_j \end{bmatrix} N \begin{bmatrix} Q_i^T & 0 \\ \\ 0 & Q_j^T \end{bmatrix} \end{aligned} \tag{6.10}$$

where

$$N \triangleq \begin{bmatrix} \Lambda_i & \sqrt{\Lambda_i} \;\; I_{n_i \times n_j} \sqrt{\Lambda_j} \\ \\ \sqrt{\Lambda_j} \;\; I_{n_j \times n_i} \sqrt{\Lambda_i} & \Lambda_j \end{bmatrix} \tag{6.11}$$

It is straightforward to verify that

$$\text{rank}\{S\} = \text{rank}\{N\} = \max\left\{\text{rank}\{S_i\}, \text{rank}\{S_j\}\right\}$$

On the other hand, the Schur complement formula yields:

$$\begin{aligned} &\text{rank}\left\{W\{\mathcal{B}_i, \mathcal{B}_i\}\right\} = \text{rank}\{O^+\} + \text{rank}\{S_i\} \\ &\text{rank}\left\{W\{\mathcal{B}_j, \mathcal{B}_j\}\right\} = \text{rank}\{O^+\} + \text{rank}\{S_j\} \\ &\text{rank}\left\{W\{\mathcal{B}_i \cup \mathcal{B}_j, \mathcal{B}_i \cup \mathcal{B}_j\}\right\} = \text{rank}\{O^+\} + \text{rank}\{S\} \end{aligned}$$

(see [20]). Combining the above equations leads to the conclusion that the rank of $W\{\mathcal{B}_i \cup \mathcal{B}_j, \mathcal{B}_i \cup \mathcal{B}_j\}$ *is upper bounded by the maximum ranks of* $W\{\mathcal{B}_i, \mathcal{B}_i\}$ *and* $W\{\mathcal{B}_j, \mathcal{B}_j\}$. *On the other hand, since* N *is positive semidefinite, it follows from* (6.10) *that* $W\{\mathcal{B}_i \cup \mathcal{B}_j, \mathcal{B}_i \cup \mathcal{B}_j\} \succeq 0$. *A simple induction concludes that the output* W^{opt} *of the matrix completion algorithm is a positive semidefinite matrix whose rank is upper bounded by* $t + 1$. *The proof is completed by noting that* W^{opt} *and* W^{ref} *share the same values on their diagonals and those off-diagonal locations corresponding to the edges of the sparsity graph* $\mathcal{G}$.

6.3 Finite-horizon Deterministic ODC Problem

The primary objective of the ODC problem is to design a structurally constrained gain K. Assume that the matrix K has l free entries to be designed. Denote these parameters as $h_1, h_2, \ldots, h_l$. To formulate the ODC problem, the space of permissible controllers can be characterized as

$$\mathcal{K} \triangleq \left\{ \sum_{i=1}^{l} h_i N_i \;\middle|\; h \in \mathbb{R}^l \right\}, \tag{6.12}$$

for some (fixed) 0-1 matrices $N_1, \ldots, N_l \in \mathbb{R}^{m\times r}$. Now, the ODC problem can be stated as follows.

Finite-Horizon ODC Problem: Minimize

$$\sum_{\tau=0}^{p} \left(x[\tau]^T Q x[\tau] + u[\tau]^T R u[\tau] \right) + \alpha \|K\|_F^2 \tag{6.13a}$$

subject to

$$x[0] = x_0 \tag{6.13b}$$

$$x[\tau+1] = Ax[\tau] + Bu[\tau] \qquad \tau = 0, 1, \ldots, p-1 \tag{6.13c}$$

$$y[\tau] = Cx[\tau] \qquad \tau = 0, 1, \ldots, p \tag{6.13d}$$

$$u[\tau] = Ky[\tau] \qquad \tau = 0, 1, \ldots, p \tag{6.13e}$$

$$K = h_1 N_1 + \ldots + h_l N_l \tag{6.13f}$$

over the variables $\{x[\tau] \in \mathbb{R}^n\}_{\tau=0}^{p}$, $\{y[\tau] \in \mathbb{R}^r\}_{\tau=0}^{p}$, $\{u[\tau] \in \mathbb{R}^m\}_{\tau=0}^{p}$, $K \in \mathbb{R}^{m\times r}$ and $h \in \mathbb{R}^l$.

6.3.1 Sparsification of ODC Problem

The finite-horizon ODC is naturally a QCQP problem. Consider an arbitrary SDP relaxation of the ODC problem and let $\mathcal{G}$ be the sparsity graph of this relaxation. Due to existence of nonzero off-diagonal elements in Q and R, certain edges (and probably cycles) may exist in the subgraphs of $\mathcal{G}$ associated with the state and input vectors $x[\tau]$ and $u[\tau]$. Under this circumstance, the treewidth of $\mathcal{G}$ could be as high as n. To understand the effect of a non-diagonal controller

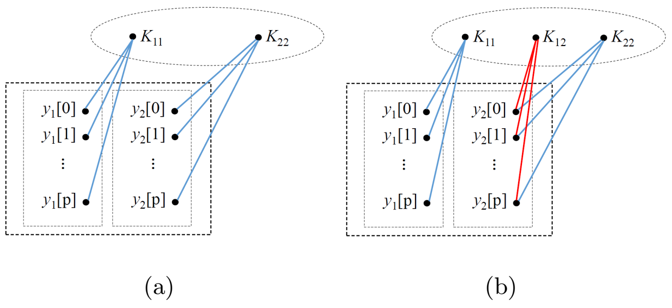


Figure 6.2: Effect of a nonzero off-diagonal entry of the controller K on the sparsity graph of the finite-horizon ODC: (a) a subgraph of $\mathcal{G}$ for the case where K_{11} and K_{22} are the only free parameters of the controller K, (b) a subgraph of $\mathcal{G}$ for the case where K_{12} is also a free parameter of the controller.

K, consider the case $m = r = 2$ and assume that the controller K under design has three free elements as follows:

$$K = \begin{bmatrix} K_{11} & K_{12} \\ 0 & K_{22} \end{bmatrix} \tag{6.14}$$

(i.e., $h_1 = K_{11}$, $h_2 = K_{12}$ and $h_3 = K_{22}$). Figure 6.2 shows a part of the graph $\mathcal{G}$. It can be observed that this subgraph is acyclic for $K_{12} = 0$ but has a cycle as soon as K_{12} becomes a free parameter. As a result, the treewidth of $\mathcal{G}$ is contingent upon the zero pattern of K. In order to guarantee existence of a low rank solution, we diagonalize Q, R and K through a reformulation of the ODC problem. Note that this transformation is redundant if Q, R and K are all diagonal.

Let $Q_d \in \mathbb{R}^{n\times n}$ and $R_d \in \mathbb{R}^{m\times m}$ be the respective eigenvector matrices of Q and R, i.e.,

$$Q = Q_d^T \Lambda_Q Q_d, , \qquad R = R_d^T \Lambda_R R_d \tag{6.15}$$

where $\Lambda_Q \in \mathbb{R}^{n\times n}$ and $\Lambda_R \in \mathbb{R}^{m\times m}$ are diagonal matrices. Notice that there exist two constant binary matrices $\Phi_1 \in \mathbb{R}^{m\times l}$ and $\Phi_2 \in \mathbb{R}^{l\times r}$ such that

$$\mathcal{K} = \left\{ \Phi_1 \text{diag}\{h\} \Phi_2 \;|\; h \in \mathbb{R}^l \right\}, \tag{6.16}$$

where diag$\{h\}$ denotes a diagonal matrix whose diagonal entries are inherited from the vector h [75]. Now, a sparse formulation of the ODC problem can be obtained in terms of the matrices

$$\bar{A} \triangleq Q_d A Q_d^T, \qquad \bar{B} \triangleq Q_d B R_d^T,$$
$$\bar{C} \triangleq \Phi_2 C Q_d^T, \qquad \bar{x}_0 \triangleq Q_d x_0,$$

and the new set of variables $\bar{x}[\tau] \triangleq Q_d x[\tau]$, $\bar{y}[\tau] \triangleq \Phi_2 y[\tau]$ and $\bar{u}[\tau] \triangleq R_d u[\tau]$ for every $\tau = 0, 1, \ldots, p$.

Reformulated Finite-Horizon ODC Problem: Minimize

$$\sum_{\tau=0}^{p} \left(\bar{x}[\tau]^T \Lambda_Q \bar{x}[\tau] + \bar{u}[\tau]^T \Lambda_R \bar{u}[\tau]\right) + \alpha \|h\|_2^2 \tag{6.17a}$$

subject to

$$\bar{x}[0] = \bar{x}_0 \times z^2 \tag{6.17b}$$
$$\bar{x}[\tau+1] = \bar{A}\bar{x}[\tau] + \bar{B}\bar{u}[\tau] \qquad \tau = 0, 1, \ldots, p-1 \tag{6.17c}$$
$$\bar{y}[\tau] = \bar{C}\bar{x}[\tau] \qquad \tau = 0, 1, \ldots, p \tag{6.17d}$$
$$\bar{u}[\tau] = R_d \Phi_1 \text{diag}\{h\} \bar{y}[\tau] \qquad \tau = 0, 1, \ldots, p \tag{6.17e}$$
$$z^2 = 1 \tag{6.17f}$$

over the variables $\{\bar{x}[\tau] \in \mathbb{R}^n\}_{\tau=0}^p$, $\{\bar{y}[\tau] \in \mathbb{R}^l\}_{\tau=0}^p$, $\{\bar{u}[\tau] \in \mathbb{R}^m\}_{\tau=0}^p$, $h \in \mathbb{R}^l$ and $z \in \mathbb{R}$.

To cast the reformulated finite-horizon ODC as a quadratic optimization, define

$$w \triangleq \begin{bmatrix} z \ h^T \ \bar{x}^T \ \bar{u}^T \ \bar{y}^T \end{bmatrix}^T \in \mathbb{R}^{n_w} \tag{6.18}$$

where $\bar{x} \triangleq \begin{bmatrix} \bar{x}[0]^T & \cdots & \bar{x}[p]^T \end{bmatrix}^T$, $\bar{u} \triangleq \begin{bmatrix} \bar{u}[0]^T & \cdots & \bar{u}[p]^T \end{bmatrix}^T$, $\bar{y} \triangleq \begin{bmatrix} \bar{y}[0]^T & \cdots & \bar{y}[p]^T \end{bmatrix}^T$ and $n_w \triangleq 1 + l + (p+1)(n+l+m)$. The scalar auxiliary variable z plays the role of number 1 (it suffices to impose the additional quadratic constraint (6.17f) as opposed to $z = 1$ without affecting the solution).

6.3.2 SDP Relaxations of ODC Problem

In this subsection, two SDP relaxations are proposed for the reformulated finite-horizon ODC problem given in (6.17). For the first relaxation, there is a guarantee on the rank of the solution.

In contrast, the second relaxation offers a tighter lower bound on the optimal cost of the ODC problem, but its solution might be high rank and therefore its rounding to a rank-1 solution could be more challenging.

6.3.2.1 Sparse SDP relaxation

Let $e_1, \ldots, e_{n_w}$ denote the standard basis for $\mathbb{R}^{n_w}$. The ODC problem consists of $n_l \triangleq (p+1)(n+l)$ linear constraints given in (6.17b), (6.17c) and (6.17d), which can be formulated as

$$D^T w = 0 \tag{6.19}$$

for some matrix $D \in \mathbb{R}^{n_w \times n_l}$. Moreover, the objective function (6.17a) and the constraints in (6.17e) and (6.17f) are all quadratic and can be expressed in terms of some matrices $M \in \mathbb{S}_{n_w}$, $\{M_i[\tau] \in \mathbb{S}_{n_w}\}_{i=1,\ldots,m;\, \tau=0,1,\ldots,p}$ and $E \triangleq e_1 e_1^T$. This leads to the following formulation of (6.17).

Sparse Formulation of ODC Problem: Minimize

$$\langle M, ww^T \rangle \tag{6.20a}$$

subject to

$$D^T w = 0 \tag{6.20b}$$

$$\langle M_i[\tau], ww^T \rangle = 0 \qquad i = 1, \ldots, m, \quad \tau = 0, 1, \ldots, p \tag{6.20c}$$

$$\langle E, ww^T \rangle = 1 \tag{6.20d}$$

with the variable $w \in \mathbb{R}^{n_w}$.

For every $j = 1, \ldots, n_l$, define

$$D_j = D_{:,j} e_j^T + e_j D_{:,j}^T \tag{6.21}$$

where $D_{:,j}$ denotes the j-th column of D. An SDP relaxation of (6.20) will be obtained below.

Sparse Relaxation of Finite-Horizon ODC: Minimize

$$\langle M, W \rangle \tag{6.22a}$$

subject to

$$\langle D_j, W\rangle = 0 \qquad j = 1, \dots, n_l \tag{6.22b}$$

$$\langle M_i[\tau], W\rangle = 0 \qquad i = 1, \dots, m, \quad \tau = 0, 1, \dots, p \tag{6.22c}$$

$$\langle E, W\rangle = 1 \tag{6.22d}$$

$$W \succeq 0 \tag{6.22e}$$

with the variable $W \in \mathbb{S}_{n_w}$.

The problem (6.22) is a convex relaxation of the QCQP problem (6.20). The sparsity graph of this problem is equal to

$$\begin{aligned}\mathcal{G} =& \mathcal{G}(D_1) \cup \dots \cup \mathcal{G}(D_{n_l}) \cup \mathcal{G}(M_1[0]) \cup \dots \cup \mathcal{G}(M_m[0]) \\ & \cup \dots \cup \mathcal{G}(M_1[p]) \cup \dots \cup \mathcal{G}(M_m[p])\end{aligned}$$

where the vertices of $\mathcal{G}$ correspond to the entries of w. In particular, the vertex set of $\mathcal{G}$ can be partitioned into five vertex subsets, where subset 1 consists of a single vertex associated with the variable z and subsets 2-5 correspond to the vectors $\bar{x}$, $\bar{u}$, $\bar{y}$ and h, respectively. The underlying sparsity graph $\mathcal{G}$ for the sparse formulation of the ODC problem is drawn in Figure 6.3, where each vertex of the graph is labeled by its corresponding variable. To maintain the readability of the graph, some edges of vertex z are not shown in the picture. Indeed, z is connected to all vertices corresponding to the elements of $\bar{x}$, $\bar{u}$ and $\bar{y}$ due to the linear terms in (6.20b).

Theorem 6.2. *The sparsity graph of the sparse relaxation of the finite-horizon ODC problem has treewidth 2.*

Proof 6.2. *It follows from the graph drawn in Figure 6.3 that removing vertex z from the sparsity graph $\mathcal{G}$ makes the remaining subgraph acyclic. This implies that the treewidth of $\mathcal{G}$ is at most 2. On the other hand, the treewidth cannot be 1 in light of the cycles of the graph.*

Consider the variable W of the SDP relaxation (6.22). The exactness of this relaxation is tantamount to the existence of an optimal rank-1 solution W^{opt} for (6.22). In this case, an optimal vector w^{opt} for the ODC problem can be recovered by decomposing W^{opt} as $(w^{\text{opt}})(w^{\text{opt}})^T$ (note that w has been defined in (6.18)). The following observation can be made.

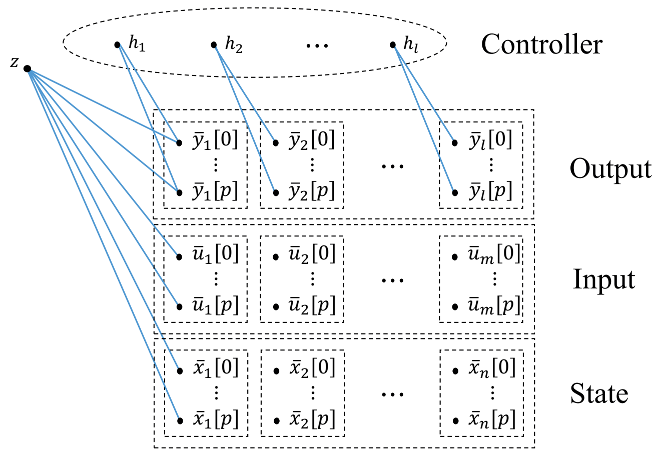


Figure 6.3: Sparsity graph of the problem (6.22) (some edges of vertex z are not shown to improve the legibility of the graph).

Corollary 6.1. *The sparse relaxation of the finite-horizon ODC problem has a matrix solution with rank at most 3.*

Proof 6.3. *This corollary is an immediate consequence of Theorems 6.1 and 6.2.*

Remark 6.2. *Since the treewidth of the sparse relaxation of the finite-horizon ODC problem* (6.22) *is equal to 2, it is possible to significantly reduce its computational complexity. More precisely, the complicating constraint $W \succeq 0$ can be replaced by positive semidefinite constraints on certain 3×3 submatrices of W, as given below:*

$$W\{\mathcal{B}_i, \mathcal{B}_i\} \succeq 0, \quad k = 1, \ldots, |\mathcal{T}| \tag{6.23}$$

where $\mathcal{T}$ is an optimal tree decomposition of the sparsity graph $\mathcal{G}$, and $\mathcal{B}_1, \ldots, \mathcal{B}_{|\mathcal{T}|}$ denote its bags. After this simplification of the hard constraint $W \succeq 0$, a quadratic number of entries of W turn out to be redundant (not appearing in any constraint) and can be removed from the optimization [44, 85].

6.3.2.2 Dense SDP relaxation

Define $D^{\perp} \in \mathbb{R}^{n_w \times (n_w - n_l)}$ as an arbitrary full row rank matrix satisfying the relation $D^T D^{\perp} = 0$. It follows from (6.20b) that every feasible vector w satisfies the equation $w = D^{\perp}\tilde{w}$, for a vector

$\tilde{w} \in \mathbb{R}^{(n_w - n_l)}$. Define

$$\tilde{M} = (D^{\perp})^T M D^{\perp} \tag{6.24a}$$

$$\tilde{M}_i[\tau] = (D^{\perp})^T M_i[\tau] D^{\perp} \tag{6.24b}$$

$$\tilde{E} = (D^{\perp})^T e_1 e_1^T D^{\perp}. \tag{6.24c}$$

The problem (6.20) can be cast in terms of $\tilde{w}$ as shown below.

Dense Formulation of ODC Problem: Minimize

$$\langle \tilde{M}, \tilde{w}\tilde{w}^T \rangle \tag{6.25a}$$

subject to

$$\langle \tilde{M}_i[\tau], \tilde{w}\tilde{w}^T \rangle = 0 \qquad i = 1, \ldots, m; \quad \tau = 0, 1, \ldots, p \tag{6.25b}$$

$$\langle \tilde{E}, \tilde{w}\tilde{w}^T \rangle = 1 \tag{6.25c}$$

over $\tilde{w} \in \mathbb{R}^{(n_w - n_l)}$.

The SDP relaxation of the above formulation is provided next.

Dense Relaxation of Finite-Horizon ODC: Minimize

$$\langle \tilde{M}, \tilde{W} \rangle \tag{6.26a}$$

subject to

$$\langle \tilde{M}_i[\tau], \tilde{W} \rangle = 0 \qquad i = 1, \ldots, m; \quad \tau = 0, 1, \ldots, p \tag{6.26b}$$

$$\langle \tilde{E}, \tilde{W} \rangle = 1 \tag{6.26c}$$

$$\tilde{W} \succeq 0 \tag{6.26d}$$

over $\tilde{W} \in \mathbb{S}_{(n_w - n_l)}$.

Remark 6.3. *Let $\mathcal{F}_s$ and $\mathcal{F}_d$ denote the feasible sets for the sparse and dense SDP relaxation problems in* (6.22) *and* (6.26)*, respectively. It can be easily seen that*

$$\{D^{\perp}\tilde{W}(D^{\perp})^T \mid \tilde{W} \in \mathcal{F}_d\} \subseteq \mathcal{F}_s \tag{6.27}$$

Therefore, the lower bound provided by the dense SDP relaxation problem (6.26) *is equal to or tighter than that of the sparse SDP relaxation* (6.22)*. However, the rank of the SDP solution of the dense relaxation may be high, which complicates its rounding to a rank-1 matrix. Hence, the sparse relaxation may be useful for recovering a near-global controller, while the dense relaxation may be used to bound the global optimality degree of the recovered controller.*

6.3.3 Rounding of SDP Solution to Rank-1 Matrix

Let W^{opt} either denote a low-rank solution for the sparse relaxation (6.22) or be equal to $D^{\perp}\tilde{W}^{\text{opt}}(D^{\perp})^T$ for a low-rank solution $\tilde{W}^{\text{opt}}$ (if any) of the dense relaxation (6.26). If the rank of W^{opt} is 1, then W^{opt} can be mapped back into a globally optimal controller for the ODC problem through an eigenvalue decomposition $W^{\text{opt}} = w^{\text{opt}}(w^{\text{opt}})^T$. Assume that W^{opt} does not have rank 1. There are multiple heuristic methods to recover a near-global controller, some of which are delineated below.

Direct Recovery Method: If W^{opt} had rank 1, then the $(2,1),(3,1),\dots,(|h|+1,1)$ entries of W^{opt} would have corresponded to the vector h^{opt} containing the free entries of K^{opt}. Inspired by this observation, if W^{opt} has rank greater than 1, then a near-global controller may still be recovered from the first column of W^{opt}. We refer to this approach as *Direct Recovery Method.*

Penalized SDP Relaxation: Recall that an SDP relaxation can be obtained by eliminating a rank constraint. In the case where this removal changes the solution, one strategy is to compensate for the rank constraint by incorporating an additive penalty function, denoted as $\Psi(W)$, into the objective of SDP relaxation. A common penalty function $\Psi(\cdot)$ is $\varepsilon \times \text{trace}\{W\}$, where ε is a design parameter. This problem is referred to as *Penalized SDP Relaxation* throughout this chapter.

Indirect Recovery Method: Define x as the aggregate state vector obtained by stacking $x[0],...,x[p]$. The objective function of every proposed SDP relaxation depends strongly on x and only weakly on k if α is small. In particular, if $\alpha = 0$, then the SDP objective function is not in terms of K. This implies that the relaxation may have two feasible matrix solutions both leading to the same optimal cost such that their first columns overlap on the part corresponding to x

and not the part corresponding to h. Hence, unlike the direct method that recovers h from the first column of W^{opt}, it may be advantageous to first recover x and then solve a second convex optimization to generate a structured controller that is able to generate state values as closely to the recovered aggregate state vector as possible. More precisely, given an SDP solution W^{opt}, define $\hat{x} \in \mathbb{R}^{n(p+1)}$ as a vector containing the entries $(|h|+2,1), (|h|+3,1), \ldots, (1+|h|+n(p+1),1)$ of W^{opt}. Define the indirect recovery method as the convex optimization problem

$$\text{minimize} \qquad \sum_{\tau=0}^{p} \|\hat{x}[\tau+1] - (A+BKC)\hat{x}[\tau]\|^2 \tag{6.28a}$$

$$\text{subject to} \qquad K = h_1 M_1 + \ldots + h_l M_l \tag{6.28b}$$

with the variables $K \in \mathbb{R}^{m\times r}$ and $h \in \mathbb{R}^l$. Let $\hat{K}$ denote a solution of the above problem. In the case where W^{opt} has rank 1 or the state part of the matrix W^{opt} corresponds to the true solution of the ODC problem, $\hat{x}$ is the same as x^{opt} and $\hat{K}$ is an optimal controller. Otherwise, $\hat{K}$ is a feasible controller that aims to make the closed-loop system follow the near-optimal state trajectory vector $\hat{x}$. As tested in [38], the above controller recovery method exhibits a remarkable performance on power systems.

6.3.4 Computationally-Cheap SDP Relaxation

Two SDP relaxations have been proposed earlier. Although these problems are convex, it may be difficult to solve them efficiently for a large-scale system. This is due to the fact that the size of each SDP matrix depends on the number of scalar variables at all times from 0 to p. There is an efficient approach to derive a computationally-cheap SDP relaxation. This will be explained below for the case where Q and R are non-singular and $r, m \le n$.

Notation 6.1. *For every matrix $M \in \mathbb{R}^{n_1\times n_2}$, define the sparsity pattern of M as follows*

$$\mathcal{S}(M) \triangleq \{S \in \mathbb{R}^{n_1\times n_2} \mid \forall(i,j)\ M_{ij} = 0 \Rightarrow S_{ij} = 0\} \tag{6.29}$$

With no loss of generality, we assume that C has full row rank. There exists an invertible matrix $\Phi \in \mathbb{R}^{n\times n}$ such that $C\Phi = \begin{bmatrix} I_r & 0 \end{bmatrix}$. Define also

$$\mathcal{K}^2 \triangleq \{\Phi_1 S \Phi_1^T \mid S \in \mathcal{S}(\Phi_2\Phi_2^T)\}. \tag{6.30}$$

Indeed, $\mathcal{K}^2$ captures the sparsity pattern of the matrix KK^T. For example, if $\mathcal{K}$ consists of block-diagonal (rectangular) matrix, $\mathcal{K}^2$ will also include block-diagonal (square) matrices. Let $\mu \in \mathbb{R}$ be a positive number such that $Q \succ \mu \times \Phi^{-T}\Phi^{-1}$, where Φ^{-T} denotes the transpose of the inverse of Φ. Define

$$\widehat{Q} := Q - \mu \times \Phi^{-T}\Phi^{-1}. \tag{6.31}$$

Using the slack matrix variables

$$X \triangleq [x[0]\ x[1]\ \dots\ x[p]]\,, \tag{6.32a}$$

$$U \triangleq [u[0]\ u[1]\ \dots\ u[p]]\,, \tag{6.32b}$$

an efficient relaxation of the ODC problem can be obtained.

Computationally-Cheap Relaxation of Finite-Horizon ODC: Minimize

$$\text{trace}\{X^T\widehat{Q}X + \mu\ \mathbf{W}_{22} + U^TRU\} + \alpha\ \text{trace}\{\mathbf{W}_{33}\} \tag{6.33a}$$

subject to

$$x[\tau+1] = Ax[\tau] + Bu[\tau], \quad \tau = 0, 1, \dots, p-1, \tag{6.33b}$$

$$x[0] = x_0, \tag{6.33c}$$

$$\mathbf{W} = \begin{bmatrix} I_n & \Phi^{-1}X\,[K \quad 0]^T \\ X^T\Phi^{-T} & \mathbf{W}_{22} & U^T \\ [K \quad 0] & U & \mathbf{W}_{33} \end{bmatrix}, \tag{6.33d}$$

$$K \in \mathcal{K}, \tag{6.33e}$$

$$\mathbf{W}_{33} \in \mathcal{K}^2, \tag{6.33f}$$

$$\mathbf{W} \succeq 0, \tag{6.33g}$$

over $K \in \mathbb{R}^{m\times r}$, $X \in \mathbb{R}^{n\times(p+1)}$, $U \in \mathbb{R}^{m\times(p+1)}$ and $\mathbf{W} \in \mathbb{S}_{n+m+p+1}$ (note that $\mathbf{W}_{22}$ and $\mathbf{W}_{33}$ are two blocks of the variable $\mathbf{W}$).

Note that the above relaxation can be naturally cast as an SDP problem by replacing each quadratic term in its objective with a new variable and then using the Schur complement. We refer to the SDP formulation of this problem as **computationally-cheap SDP relaxation**.

Theorem 6.3. *The problem* (6.33) *is a convex relaxation of the ODC problem. Furthermore, the relaxation is exact if and only if it possesses a solution* $(K^{opt}, X^{opt}, U^{opt}, \mathbf{W}^{opt})$ *such that* $\text{rank}\{\mathbf{W}^{opt}\} = n$.

Proof 6.4. *It is evident that the problem* (6.33) *is a convex program. To prove the remaining parts of the theorem, it suffices to show that the ODC problem is equivalent to* (6.33) *subject to the additional constraint rank*$\{\mathbf{W}\} = n$*. To this end, consider a feasible solution* $(K, X, U, \mathbf{W})$ *such that rank*$\{\mathbf{W}\} = n$*. Since* I_n *has rank* n*, taking the Schur complement of the blocks* $(1,1)$, $(1,2)$, $(2,1)$ *and* $(2,2)$ *of* $\mathbf{W}$ *yields that*

$$0 = \mathbf{W}_{22} - X^T \Phi^{-T} (I_n)^{-1} \Phi^{-1} X \tag{6.34}$$

Likewise,

$$0 = \mathbf{W}_{33} - KK^T \tag{6.35}$$

On the other hand,

$$\sum_{\tau=0}^{p} \left(x[\tau]^T Q x[\tau] + u[\tau]^T R u[\tau] \right) = \mathit{trace}\{X^T Q X + U^T R U\} \tag{6.36}$$

It follows from (6.34), (6.35) *and* (6.36) *that the ODC problem and its computationally cheap relaxation lead to the same objective at the respective points* (K, X, U) *and* $(K, X, U, \mathbf{W})$*. In addition, it can be concluded from the Schur complement of the blocks* $(1,1)$, $(1,2)$, $(3,1)$ *and* $(3,2)$ *of* $\mathbf{W}$ *that*

$$U = [K \quad 0] \Phi^{-1} X = KCX \tag{6.37}$$

or equivalently

$$u[\tau] = KCx[\tau] \qquad \text{for} \quad \tau = 0, 1, \ldots, p \tag{6.38}$$

This implies that (K, X, U) *is a feasible solution of the ODC problem. Hence, the optimal objective value of the ODC problem is a lower bound on that of the computationally-cheap relaxation under the additional constraint rank*$\{\mathbf{W}\} = n$.

Now, consider a feasible solution (K, X, U) of the ODC problem. Define $\mathbf{W}_{22} = X^T \Phi^{-T} \Phi^{-1} X$ and $K_2 = KK^T$. Observe that $\mathbf{W}$ can be written as the rank-n matrix $W_r W_r^T$, where

$$W_r = \begin{bmatrix} I_n \, \Phi^{-1} X \, [K \quad 0]^T \end{bmatrix}^T \tag{6.39}$$

Thus, $(K, X, U, \mathbf{W})$ is a feasible solution of the computationally-cheap SDP relaxation. This implies that the optimal objective value of the ODC problem is an upper bound on that of the computationally-cheap SDP relaxation under the additional constraint rank$\{\mathbf{W}\} = n$. The proof is completed by combining this property with its opposite statement proved earlier.

The sparse and dense SDP relaxations were both obtained by defining a matrix W as the product of two *vectors*. However, the computationally-cheap relaxation of the finite-horizon ODC Problem is obtained by defining $\mathbf{W}$ as the product of two *matrices*. This significantly reduces the computational complexity. To shed light on this fact, notice that the numbers of rows for the matrix variables of sparse and dense SDP relaxations are on the order of np, whereas the number of rows for the computationally-cheap SDP solution is on the order of $n + p$.

Remark 6.4. *The computationally-cheap relaxation of the finite-horizon ODC Problem automatically acts as a* <u>*penalized*</u> *SDP relaxation. To explain this remarkable feature of the proposed relaxation, notice that the terms* trace$\{\mathbf{W}_{22}\}$ *and* trace$\{\mathbf{W}_{33}\}$ *in the objective function of the relaxation inherently penalize the trace of $\mathbf{W}$. This automatic penalization helps significantly with the reduction of the rank of $\mathbf{W}$ at optimality. As a result, it is expected that the quality of the relaxation will be better for higher values of α and μ.*

Remark 6.5. *Consider the extreme case where $r = n$, $C = I_n$, $\alpha = 0$, $p = \infty$, and the unknown controller K is unstructured. This amounts to the famous LQR problem and the optimal controller can be found using the Riccati equation. It is straightforward to verify that the computationally-cheap relaxation of the ODC problem is always exact in this case even though it is infinite-dimensional. The proof is based on the following facts:*

- *When K is unstructured, the constraint* (6.33e) *and* (6.33f) *can be omitted. Therefore, there is no structural constraint on $\mathbf{W}_{33}$ and $\mathbf{W}_{31}$ (i.e., the $(3, 1)$ block entry).*

- *Then, the constraint* (6.33d) *reduces to* $\mathbf{W}_{22} = X^T\Phi^{-T}\Phi^{-1}X$ *due to the term* $\text{trace}\{\mathbf{W}_{22}\}$ *in the objective function. Consequently, the objective function can be rearranged as* $\sum_{\tau=0}^{\infty}\left(x[\tau]^T Q x[\tau] + u[\tau]^T R u[\tau]\right)$.
- *The only remaining constraints are the state evolution equation and* $x[0] = x_0$. *It is known that the remaining feed-forward problem has a solution* (X^{opt}, U^{opt}) *such that* $U^{opt} = K^{opt}X^{opt}$ *for some matrix* K^{opt}.

6.3.5 Stability Enforcement

The finite-horizon ODC problem studied before had no stability conditions. It is verified in some simulations in [38] that the closed-loop stability may be automatically guaranteed for physical systems, provided p is large enough. In this subsection, we aim to directly enforce stability by imposing additional constraints on the proposed SDP relaxations.

Theorem 6.4. *There exists a controller* $u[\tau] = Ky[\tau]$ *with the structure* $K \in \mathcal{K}$ *to stabilize the system* (6.1) *if and only if there exist a (Lyapunov) matrix* $P \in \mathbb{S}_n$*, a matrix* $K \in \mathbb{R}^{m\times r}$*, and auxiliary variables* $L \in \mathbb{R}^{m\times n}$ *and* $\mathbf{G} \in \mathbb{S}_{2n+m}$ *such that*

$$\begin{bmatrix} P - I_n & AP + B\mathbf{G}_{32} \\ PA^T + \mathbf{G}_{23}B^T & P \end{bmatrix} \succeq 0, \tag{6.40a}$$

$$K \in \mathcal{K}, \tag{6.40b}$$

$$\mathbf{G} \succeq 0, \tag{6.40c}$$

$$\mathbf{G}_{33} \in \mathcal{K}^2, \tag{6.40d}$$

$$\text{rank}\{\mathbf{G}\} = n, \tag{6.40e}$$

where

$$\mathbf{G} \triangleq \begin{bmatrix} I_n & \Phi^{-1}P\,[K \quad 0]^T \\ P\Phi^{-T} & \mathbf{G}_{22} & \mathbf{G}_{23} \\ [K \quad 0] & \mathbf{G}_{32} & \mathbf{G}_{33} \end{bmatrix} \tag{6.41}$$

Proof 6.5. *It is well-known that the system* (6.1) *is stable under a controller* $u[\tau] = Ky[\tau]$ *if and only if there exists a positive-definite matrix* $P \in \mathbb{S}_n$ *to satisfy the Lyapunov inequality:*

$$(A+BKC)^T P(A+BKC) - P + I_n \preceq 0 \tag{6.42}$$

or equivalently

$$\begin{bmatrix} P - I_n & AP + BKCP \\ PA^T + PK^TC^TB^T & P \end{bmatrix} \succeq 0 \tag{6.43}$$

Due to the analogy between $\mathbf{W}$ *and* $\mathbf{G}$*, the argument made in the proof of Theorem 6.3 can be adopted to complete the proof of this theorem (note that* $\mathbf{G}_{32}$ *plays the role of* KCP*).*

Theorem 6.4 translates the stability of the closed-loop system into a rank-n condition. Consider one of the aforementioned SDP relaxations of the ODC problem. To enforce stability, it results from Theorem 6.4 that two actions can be taken: (i) addition of the convex constraints (6.40a)-(6.40d) to SDP relaxations, (ii) compensation for the rank-n condition through an appropriate convex penalization of $\mathbf{G}$ in the objective function of SDP relaxations. Note that the penalization is vital because otherwise $\mathbf{G}_{22}$ and $\mathbf{G}_{33}$ would grow unboundedly to satisfy the condition $\mathbf{G} \succeq 0$.

6.4 Infinite-horizon Deterministic ODC Problem

In this section, we study the infinite-horizon ODC problem, corresponding to $p = +\infty$ and subject to a stability condition.

6.4.1 Lyapunov Formulation

The finite-horizon ODC was investigated through a time-domain formulation. However, to deal with the infinite dimension of the infinite-horizon ODC and its hard stability constraint, a Lyapunov approach will be taken here. Consider the following optimization problem.

Lyapunov Formulation of ODC: Minimize

$$x_0^T P x_0 + \alpha \|K\|_F^2 \tag{6.44a}$$

subject to

$$\begin{bmatrix} G & G & (AG+BL)^T & L^T \\ G & Q^{-1} & 0 & 0 \\ AG+BL & 0 & G & 0 \\ L & 0 & 0 & R^{-1} \end{bmatrix} \succeq 0, \tag{6.44b}$$

$$\begin{bmatrix} P & I_n \\ I_n & G \end{bmatrix} \succeq 0, \tag{6.44c}$$

$$K \in \mathcal{K}, \tag{6.44d}$$

$$L = KCG, \tag{6.44e}$$

over $K \in \mathbb{R}^{m\times r}$, $L \in \mathbb{R}^{m\times n}$, $P \in \mathbb{S}_n$ and $G \in \mathbb{S}_n$.

It will be shown in the next theorem that the above formulation is tantamount to the infinite-horizon ODC problem.

Theorem 6.5. *The infinite-horizon deterministic ODC problem is equivalent to finding a controller $K \in \mathcal{K}$, a symmetric Lyapunov matrix $P \in \mathbb{S}_n$, an auxiliary symmetric matrix $G \in \mathbb{S}_n$ and an auxiliary matrix $L \in \mathbb{R}^{m\times n}$ to solve the optimization problem* (6.44)*.*

Proof 6.6. *Given an arbitrary control gain K, we have:*

$$\sum_{\tau=0}^{\infty} \left(x[\tau]^T Q x[\tau] + u[\tau]^T R u[\tau] \right) = x[0]^T P x[0] \tag{6.45}$$

where

$$P = (A+BKC)^T P (A+BKC) + Q + (KC)^T R (KC) \tag{6.46a}$$

$$P \succeq 0 \tag{6.46b}$$

On the other hand, it is well-known that replacing the equality sign "=" in (6.46a) *with the inequality sign "$\succeq$" does not affect the solution of the optimization problem [18]. After pre- and* *post-multiplying the Lyapunov inequality obtained from* (6.46a) *with* P^{-1} *and using the Schur complement formula, the constraints* (6.46a) *and* (6.46b) *can be combined as*

$$\begin{bmatrix} P^{-1} & P^{-1} & S^T & P^{-1}(KC)^T \\ P^{-1} & Q^{-1} & 0 & 0 \\ S & 0 & P^{-1} & 0 \\ (KC)P^{-1} & 0 & 0 & R^{-1} \end{bmatrix} \succeq 0 \tag{6.47}$$

where $S = (A + BKC)P^{-1}$*. By replacing* P^{-1} *with a new variable* G *in the above matrix and defining* L *as* KCG*, the constraints* (6.44b) *and* (6.44e) *will be obtained. On the other hand,* (6.44c) *implies that* $G \succ 0$ *and* $P \succeq G^{-1}$ *. Therefore, the minimization of* $x_0^T P x_0$ *subject to the constraint* (6.44c) *ensures that* $P = G^{-1}$ *is satisfied for at least one optimal solution of the optimization problem.*

Theorem 6.6. *Consider the special case where* $r = n$, $C = I_n$, $\alpha = 0$ *and* $\mathcal{K}$ *contains the set of all unstructured controllers. Then, the infinite-horizon deterministic ODC problem has the same solution as the convex optimization problem obtained from the nonlinear optimization* (6.44) *by removing its non-convex constraint* (6.44e).

Proof 6.7. *It is easy to verify that a solution* $(K^{opt}, P^{opt}, G^{opt}, L^{opt})$ *of the convex problem stated in the theorem can be mapped to the solution* $(L^{opt}(G^{opt})^{-1}, P^{opt}, G^{opt}, L^{opt})$ *of the non-convex problem* (6.44) *and vice versa (recall that* $C = I_n$ *by assumption). This completes the proof.*

6.4.2 SDP Relaxation

Theorem 6.6 states that a classical optimal control problem can be precisely solved via a convex relaxation of the nonlinear optimization (6.44) by eliminating its constraint (6.44e). However, this simple convex relaxation does not work satisfactorily for a general control structure $K = \Phi_1 \text{diag}\{h\}\Phi_2$. To design a better relaxation, define

$$w = \begin{bmatrix} 1 & h^T & \text{vec}\{\Phi_2 C G\}^T \end{bmatrix}^T \tag{6.48}$$

where $\mathrm{vec}\{\Phi_2 CG\}$ is an $nl \times 1$ column vector obtained by stacking the columns of $\Phi_2 CG$. It is possible to write every entry of the bilinear matrix term KCG as a linear function of the entries of the parametric matrix ww^T. Hence, by introducing a new matrix variable $\mathbf{W}$ playing the role of ww^T, the nonlinear constraint (6.44e) can be rewritten as a linear constraint in term of $\mathbf{W}$.

Notation 6.2. *Define the sampling operator* $\mathrm{samp} : \mathbb{R}^{l\times nl} \to \mathbb{R}^{l\times n}$ *as follows:*

$$\mathrm{samp}\{X\} = \left[X_{i,(n-1)j+i}\right]_{i=1,\dots,l;\ j=1,\dots,n}. \tag{6.49}$$

Now, one can relax the non-convex mapping constraint $\mathbf{W} = ww^T$ to $\mathbf{W} \succeq 0$ and another constraint stating that the first column of $\mathbf{W}$ is equal to w. This yields the following convex relaxation of problem (6.44).

SDP Relaxation of Infinite-Horizon Deterministic ODC: Minimize

$$x_0^T P x_0 + \alpha \ \mathrm{trace}\{\mathbf{W}_{33}\} \tag{6.50a}$$

subject to

$$\begin{bmatrix} G & G & (AG+BL)^T & L^T \\ G & Q^{-1} & 0 & 0 \\ AG+BL & 0 & G & 0 \\ L & 0 & 0 & R^{-1} \end{bmatrix} \succeq 0, \tag{6.50b}$$

$$\begin{bmatrix} P & I_n \\ I_n & G \end{bmatrix} \succeq 0, \tag{6.50c}$$

$$L = \Phi_1 \times \text{samp}\{\mathbf{W}_{32}\}, \tag{6.50d}$$

$$\mathbf{W} = \begin{bmatrix} 1 & \text{vec}\{\Phi_2 CG\}^T & h^T \\ \text{vec}\{\Phi_2 CG\} & \mathbf{W}_{22} & \mathbf{W}_{23} \\ h & \mathbf{W}_{32} & \mathbf{W}_{33} \end{bmatrix}, \tag{6.50e}$$

$$\mathbf{W} \succeq 0, \tag{6.50f}$$

over $h \in \mathbb{R}^l$, $L \in \mathbb{R}^{m\times n}$, $P \in \mathbb{S}_n$, $G \in \mathbb{S}_n$ and $\mathbf{W} \in \mathbb{S}_{1+l(n+1)}$.

If the relaxed problem (6.50) has the same solution as the infinite-horizon ODC in (6.44), the relaxation is exact.

Theorem 6.7. *The following statements hold regarding the relaxation of the infinite-horizon deterministic ODC in* (6.50)*:*

i) The relaxation is exact if it has a solution $(h^{opt}, P^{opt}, G^{opt}, L^{opt}, \mathbf{W}^{opt})$ *such that* $\text{rank}\{\mathbf{W}^{opt}\} = 1$.

ii) The relaxation always has a solution $(h^{opt}, P^{opt}, G^{opt}, L^{opt}, \mathbf{W}^{opt})$ *such that* $\text{rank}\{\mathbf{W}^{opt}\} \leq 3$.

Proof 6.8. *Consider a sparsity graph $\mathcal{G}$ of* (6.50)*, constructed as follows. The graph $\mathcal{G}$ has $1+l(n+1)$ vertices corresponding to the rows of $\mathbf{W}$. Two arbitrary disparate vertices $i, j \in \{1, 2, \ldots, 1+l(n+1)\}$ are adjacent in $\mathcal{G}$ if $\mathbf{W}_{ij}$ appears in at least one of the constraints of the problem* (6.50) *excluding the global constraint $\mathbf{W} \succeq 0$. For example, vertex 1 is connected to all remaining vertices of $\mathcal{G}$. The graph $\mathcal{G}$ with its vertex 1 removed is depicted in Figure 6.4. This graph is acyclic and therefore the treewidth of $\mathcal{G}$ is at most* 2*. Hence, it follows from Theorem 1 that* (6.50) *has a matrix solution with rank at most* 3*.*

Theorem 6.7 states that the SDP relaxation of the infinite-horizon ODC problem has a low-rank solution. However, it does not imply that every solution of the relaxation is low-rank. Theorem 1 provides a procedure for converting a high-rank solution of the SDP relaxation into a low-rank one.

6.4.3 Computationally-Cheap Relaxation

The aforementioned SDP relaxation has a high dimension for a large-scale system, which makes it less interesting for computational purposes. Moreover, the quality of its optimal objective value can be improved using some indirect penalty technique. The objective of this subsection is to offer a computationally-cheap SDP relaxation for the ODC problem, whose solution outperforms that of the previous SDP relaxation. For this purpose, consider again a scalar μ such that $Q \succ \mu \times \Phi^{-T}\Phi^{-1}$ and define $\hat{Q}$ according to (6.31).

Computationally-Cheap Relaxation of Infinite-horizon Deterministic ODC: Minimize

$$x_0^T P x_0 + \alpha \ \text{trace}\{\mathbf{W}_{33}\} \tag{6.51a}$$

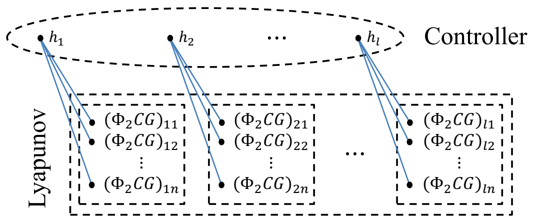


Figure 6.4: The sparsity graph for the infinite-horizon deterministic ODC in the case where $\mathcal{K}$ consists of diagonal matrices (the central vertex corresponding to the constant 1 is removed for simplicity).

subject to

$$\begin{bmatrix} G-\mu\mathbf{W}_{22} & G & (AG+BL)^T & L^T \\ G & \widehat{Q}^{-1} & 0 & 0 \\ AG+BL & 0 & G & 0 \\ L & 0 & 0 & R^{-1} \end{bmatrix} \succeq 0, \tag{6.51b}$$

$$\begin{bmatrix} P & I_n \\ I_n & G \end{bmatrix} \succeq 0, \tag{6.51c}$$

$$\mathbf{W} = \begin{bmatrix} I_n & \Phi^{-1}G\,[K \quad 0]^T \\ G\Phi^{-T} & \mathbf{W}_{22} & L^T \\ [K \quad 0] & L & \mathbf{W}_{33} \end{bmatrix}, \tag{6.51d}$$

$$K \in \mathcal{K}, \tag{6.51e}$$

$$\mathbf{W}_{33} \in \mathcal{K}^2, \tag{6.51f}$$

$$\mathbf{W} \succeq 0, \tag{6.51g}$$

over $K \in \mathbb{R}^{m\times r}$, $L \in \mathbb{R}^{m\times n}$, $P \in \mathbb{S}_n$, $G \in \mathbb{S}_n$ and $\mathbf{W} \in \mathbb{S}_{2n+m}$.

The following remarks can be made regarding (6.51):

- The constraint (6.51b) corresponds to the Lyapunov inequality associated with (6.46a), where $\mathbf{W}_{22}$ in its first block aims to play the role of $P^{-1}\Phi^{-T}\Phi^{-1}P^{-1}$.
- The constraint (6.51c) ensures that the relation $P = G^{-1}$ occurs at optimality (at least for one of the solution of the problem).
- The constraint (6.51d) is a surrogate for the only complicating constraint of the ODC problem, i.e., $L = KCG$.
- Since no non-convex rank constraint is imposed on the problem to maintain the convexity of the relaxation, the rank constraint is compensated in various ways. More precisely, the entries of $\mathbf{W}$ are constrained in the objective function (6.51a) through the term α trace$\{\mathbf{W}_{33}\}$, in the first block of the constraint (6.51b) through the term $G - \mu\mathbf{W}_{22}$, and also via the constraint (6.51e) and (6.51f). These terms aim to automatically penalize the rank of $\mathbf{W}$ indirectly.
- The proposed relaxation takes advantage of the sparsity of not only K, but also KK^T (through the constraint (6.51f)).

Theorem 6.8. *The problem* (6.51) *is a convex relaxation of the infinite-horizon ODC problem. Furthermore, the relaxation is exact if and only if it possesses a solution* $(K^{opt}, L^{opt}, P^{opt}, G^{opt}, \mathbf{W}^{opt})$ *such that rank*$\{\mathbf{W}^{opt}\} = n$.

Proof 6.9. *The objective function and constraints of the problem* (6.51) *are all linear functions of the tuple* $(K, L, P, G, \mathbf{W})$*. Hence, this relaxation is indeed convex. To study the relationship between this optimization problem and the infinite-horizon ODC, consider a feasible point* (K, L, P, G) *of the ODC formulation* (6.44)*. It can be deduced from the relation* $L = KCG$ *that* $(K, L, P, G, \mathbf{W})$ *is a feasible solution of the problem* (6.51) *if the free blocks of* $\mathbf{W}$ *are considered as*

$$\mathbf{W}_{22} = G\Phi^{-T}\Phi^{-1}G, \qquad \mathbf{W}_{33} = KK^T \tag{6.52}$$

(note that (6.44b) *and* (6.51b) *are equivalent for this choice of* $\mathbf{W}$*). This implies that the problem* (6.51) *is a convex relaxation of the infinite-horizon ODC problem.*

Consider now a solution $(K^{opt}, L^{opt}, P^{opt}, G^{opt}, \mathbf{W}^{opt})$ *of the computationally-cheap SDP relaxation such that rank*$\{\mathbf{W}^{opt}\} = n$. *Since the rank of the first block of* $\mathbf{W}^{opt}$ *(i.e.,* I_n*) is already* n*, a Schur complement argument on the blocks* $(1,1)$*,* $(1,3)$*,* $(2,1)$ *and* $(2,3)$ *of* $\mathbf{W}^{opt}$ *yields that*

$$0 = L^{opt} - [K^{opt} \quad 0](I_n)^{-1}\Phi^{-1}G^{opt} \tag{6.53}$$

or equivalently $L^{opt} = K^{opt}CG^{opt}$*, which is tantamount to the constraint* (6.44e)*. This implies that* $(K^{opt}, L^{opt}, P^{opt}, G^{opt})$ *is a solution of the infinite-horizon ODC problem* (6.44) *and hence the relaxation is exact. So far, we have shown that the existence of a rank-n solution* $\mathbf{W}^{opt}$ *guarantees the exactness of the relaxation. The converse of this statement can also be proved similarly.*

The variable $\mathbf{W}$ in the first SDP relaxation (6.50) had $1 + l(n+1)$ rows. In contrast, this number reduces to $2n + m$ for the matrix $\mathbf{W}$ in the computationally-cheap relaxation (6.51). This significantly reduces the computation time of the relaxation.

Corollary 6.2. *Consider the special case where* $r = n$*,* $C = I_n$*,* $\alpha = 0$ *and* $\mathcal{K}$ *contains the set of all unstructured controllers. Then, the computationally-cheap relaxation problem* (6.51) *is exact for the infinite-horizon ODC problem.*

Proof 6.10. *The proof follows from that of Theorem 6.6.*

6.4.4 Controller Recovery

In this subsection, two controller recovery methods will be described. With no loss of generality, our focus will be on the computationally-cheap relaxation problem (6.51).

Direct Recovery Method for Infinite-Horizon ODC: A near-optimal controller K for the infinite-horizon ODC problem is chosen to be equal to the optimal matrix K^{opt} obtained from the computationally-cheap relaxation problem (6.51).

Indirect Recovery Method for Infinite-Horizon ODC: Let $(K^{\mathrm{opt}}, L^{\mathrm{opt}}, P^{\mathrm{opt}}, G^{\mathrm{opt}}, \mathbf{W}^{\mathrm{opt}})$ denote a solution of the computationally-cheap relaxation problem (6.51). Given a pre-specified nonnegative number ε, a near-optimal controller $\hat{K}$ for the infinite-horizon ODC problem is recovered by minimizing

$$\varepsilon \times \gamma + \alpha \|K\|_F^2 \tag{6.54a}$$

subject to

$$\begin{bmatrix} (G^{\text{opt}})^{-1} - Q + \gamma I_n & (A+BKC)^T & (KC)^T \\ (A+BKC) & G^{\text{opt}} & 0 \\ (KC) & 0 & R^{-1} \end{bmatrix} \succ 0 \tag{6.54b}$$

$$K = h_1 N_1 + \ldots + h_l N_l. \tag{6.54c}$$

over $K \in \mathbb{R}^{m\times r}$, $h \in \mathbb{R}^l$ and $\gamma \in \mathbb{R}$. Note that this is a convex program. The direct recovery method assumes that the controller K^{opt} obtained from the computationally-cheap relaxation problem (6.51) is near-optimal, whereas the indirect method assumes that the controller K^{opt} might be unacceptably imprecise while the inverse of the Lyapunov matrix is near-optimal. The indirect method is built on SDP relaxation by fixing G at its optimal value and then perturbing Q as $Q - \gamma I_n$ to facilitate the recovery of a stabilizing controller. The underlying idea is that the SDP relaxation depends strongly on G and weakly on P (note that G appears 9 times in the formulation, while P appears only twice to indirectly account for the inverse of G). In other words, there might be two feasible solutions with similar costs for the SDP relaxation whose G parts are identical while their P parts are very different. Hence, the indirect method focuses on G.

6.5 Infinite-Horizon Stochastic ODC Problem

This section is mainly concerned with the stochastic optimal distributed control (SODC) problem, which aims to design a stabilizing static controller $u[\tau] = Ky[\tau]$ to minimize the cost function

$$\lim_{\tau\to+\infty} \mathcal{E}\left(x[\tau]^T Q x[\tau] + u[\tau]^T R u[\tau]\right) + \alpha \|K\|_F^2 \tag{6.55}$$

subject to the system dynamics (6.3) and the controller requirement $K \in \mathcal{K}$, for a nonnegative scalar α and positive-definite matrices Q and R. Define two covariance matrices as

$$\Sigma_d = \mathcal{E}\{Ed[0]d[0]^T E^T\} \quad \Sigma_v = \mathcal{E}\{Fv[0]v[0]^T F^T\} \tag{6.56}$$

Consider the following optimization problem.

Lyapunov Formulation of SODC: Minimize

$$\langle P, \Sigma_d\rangle + \langle M + K^T RK, \Sigma_v\rangle + \alpha\|K\|_F^2 \tag{6.57a}$$

subject to

$$\begin{bmatrix} G & G & (AG+BL)^T & L^T \\ G & Q^{-1} & 0 & 0 \\ AG+BL & 0 & G & 0 \\ L & 0 & 0 & R^{-1} \end{bmatrix} \succeq 0, \tag{6.57b}$$

$$\begin{bmatrix} P & I_n \\ I_n & G \end{bmatrix} \succeq 0, \tag{6.57c}$$

$$\begin{bmatrix} M & (BK)^T \\ BK & G \end{bmatrix} \succeq 0, \tag{6.57d}$$

$$K \in \mathcal{K} \tag{6.57e}$$

$$L = KCG \tag{6.57f}$$

over the controller $K \in \mathbb{R}^{m\times r}$, Lyapunov matrix $P \in \mathbb{S}_n$ and auxiliary matrices $G \in \mathbb{S}_n$, $L \in \mathbb{R}^{m\times n}$ and $M \in \mathbb{S}_r$.

Theorem 6.9. *The infinite-horizon SODC problem adopts the non-convex formulation* (6.57).

Proof 6.11. *It is straightforward to verify that*

$$\begin{aligned} x[\tau] = {} & (A+BKC)^{\tau}x[0] \\ & + \sum_{t=0}^{\tau-1}(A+BKC)^{\tau-t-1}(Ed[t]+BKFv[t]) \end{aligned} \tag{6.58}$$

for $\tau = 1, 2, \dots, \infty$*. On the other hand, since the controller under design must be stabilizing,* $(A+BKC)^{\tau}$ *approaches zero as* τ *goes to* $+\infty$*. In light of the above equation, it can be verified that*

$$\begin{aligned} & \mathcal{E}\left\{\lim_{\tau\to+\infty}\left(x[\tau]^T Qx[\tau] + u[\tau]^T Ru[\tau]\right)\right\} \\ & = \mathcal{E}\left\{\lim_{\tau\to+\infty} x[\tau]^T\left(Q + C^TK^TRKC\right)x[\tau]\right\} \\ & \quad + \mathcal{E}\left\{\lim_{\tau\to+\infty} v[\tau]^T F^T K^T RKFv[\tau]\right\} \\ & = \langle P, \Sigma_d\rangle + \langle (BK)^T P(BK) + K^TRK, \Sigma_v\rangle \end{aligned} \tag{6.59}$$

where

$$P = \sum_{t=0}^{\infty}\left((A+BKC)^t\right)^T(Q + C^TK^TRKC)(A+BKC)^t$$

Similar to the proof of Theorem 6.5, the above infinite series can be replaced by the expanded Lyapunov inequality (6.47)*: After replacing* P^{-1} *and* KCP^{-1} *in* (6.47) *with new variables* G *and* L*, it can be concluded that:*

- *The condition* (6.47) *is identical to the set of constraints* (6.57b) *and* (6.57f)*.*
- *The cost function* (6.59) *can be expressed as*

$$\langle P, \Sigma_d\rangle + \langle (BK)^T G^{-1}(BK) + K^TRK, \Sigma_v\rangle + \alpha\|K\|_F^2$$

- *Since* P *appears only once in the constraints of the optimization problem* (6.57) *(i.e., the condition* (6.57c)*) and the objective function of this optimization includes the term* $\langle P, \Sigma_d\rangle$*, an optimal value of* P *is equal to* G^{-1} *(Notice that* $\Sigma_d \succeq 0$*).*
- *Similarly, the optimal value of* M *is equal to* $(BK)^T G^{-1}(BK)$*.*

The proof follows from the above observations.

The traditional H_2 optimal control problem (i.e., in the centralized case) can be solved using Riccati equations. It will be shown in the next proposition that dropping the nonconvex constraint (6.57f) results in a convex optimization that correctly solves the centralized H_2 optimal control problem.

Proposition 6.1. *Consider the special case where $r = n$, $C = I_n$, $\alpha = 0$, $\Sigma_v = 0$, and $\mathcal{K}$ contains the set of all unstructured controllers. Then, the SODC problem has the same solution as the convex optimization problem obtained from the nonlinear optimization* (6.57a)-(6.57) *by removing its non-convex constraint* (6.57f).

Proof 6.12. *It is similar to the proof of Theorem 6.6.*

Consider the vector w defined in (6.48). Similar to the infinite-horizon ODC case, the bilinear matrix term KCG can be represented as a linear function of the entries of the parametric matrix $\mathbf{W}$ defined as ww^T. Now, a convex relaxation can be attained by relaxing the constraint $\mathbf{W} = ww^T$ to $\mathbf{W} \succeq 0$ and adding another constraint stating that the first column of $\mathbf{W}$ is equal to w.

Relaxation of Infinite-Horizon SODC: Minimize

$$\langle P, \Sigma_d \rangle + \langle M + K^T R K, \Sigma_v \rangle + \alpha \operatorname{trace}\{\mathbf{W}_{33}\} \tag{6.60a}$$

subject to

$$\begin{bmatrix} G & G & (AG+BL)^T & L^T \\ G & Q^{-1} & 0 & 0 \\ AG+BL & 0 & G & 0 \\ L & 0 & 0 & R^{-1} \end{bmatrix} \succeq 0, \tag{6.60b}$$

$$\begin{bmatrix} P & I_n \\ I_n & G \end{bmatrix} \succeq 0, \tag{6.60c}$$

$$K = \Phi_1 \text{diag}\{h\} \Phi_2, \tag{6.60d}$$

$$\begin{bmatrix} M & (BK)^T \\ BK & G \end{bmatrix} \succeq 0, \tag{6.60e}$$

$$L = \Phi_1 \text{samp}\{\mathbf{W}_{32}\}, \tag{6.60f}$$

$$\mathbf{W} = \begin{bmatrix} 1 & \text{vec}\{\Phi_2 CG\}^T & h^T \\ \text{vec}\{\Phi_2 CG\} & \mathbf{W}_{22} & \mathbf{W}_{23} \\ h & \mathbf{W}_{32} & \mathbf{W}_{33} \end{bmatrix}, \tag{6.60g}$$

$$\mathbf{W} \succeq 0, \tag{6.60h}$$

over the controller $K \in \mathbb{R}^{m\times r}$, Lyapunov matrix $P \in \mathbb{S}_n$ and auxiliary matrices $G \in \mathbb{S}_n$, $L \in \mathbb{R}^{m\times n}$, $M \in \mathbb{S}_r$, $h \in \mathbb{R}^l$ and $\mathbf{W} \in \mathbb{S}_{1+l(n+1)}$.

Theorem 6.10. *The following statements hold regarding the convex relaxation of the infinite-horizon SODC problem:*

i) The relaxation is exact if it has a solution $(h^{opt}, K^{opt}, P^{opt}, G^{opt}, L^{opt}, M^{opt}, \mathbf{W}^{opt})$ *such that* $\text{rank}\{W^{opt}\} = 1$.

ii) The relaxation always has a solution $(h^{opt}, K^{opt}, P^{opt}, G^{opt}, L^{opt}, M^{opt}, \mathbf{W}^{opt})$ *such that* $\text{rank}\{W^{opt}\} \leq 3$.

Proof 6.13. *The proof is omitted (see Theorems 6.7 and 6.9).*

As before, it can be deduced from Theorem 6.10 that the infinite-horizon SODC problem has a convex relaxation with the property that its exactness amounts to the existence of a rank-1 matrix solution $\mathbf{W}^{\text{opt}}$. Moreover, it is always guaranteed that this relaxation has a solution such that $\text{rank}\{\mathbf{W}^{\text{opt}}\} \leq 3$.

A computationally-cheap SDP relaxation for the SODC problem will be derived below. Let μ_1 and μ_2 be two nonnegative numbers such that

$$Q \succ \mu_1 \times \Phi^{-T}\Phi^{-1}, \quad \Sigma_v \succeq \mu_2 \times I_r \tag{6.61}$$

Define $\widehat{Q} := Q - \mu_1 \times \Phi^{-T}\Phi^{-1}$ and $\widehat{\Sigma}_v := \Sigma_v - \mu_2 \times I_r$.

Computationally-Cheap Relaxation of Infinite-Horizon SODC: Minimize

$$\langle P, \Sigma_d \rangle + \langle M, \Sigma_v \rangle + \langle K^T R K, \widehat{\Sigma}_v \rangle + \langle \mu_2 R + \alpha I_m, \mathbf{W}_{33} \rangle \tag{6.62a}$$

subject to

$$\begin{bmatrix} G-\mu_1\mathbf{W}_{22} & G & (AG+BL)^T & L^T \\ G & \widehat{Q}^{-1} & 0 & 0 \\ AG+BL & 0 & G & 0 \\ L & 0 & 0 & R^{-1} \end{bmatrix} \succeq 0, \tag{6.62b}$$

$$\begin{bmatrix} P & I_n \\ I_n & G \end{bmatrix} \succeq 0, \tag{6.62c}$$

$$\begin{bmatrix} M & (BK)^T \\ BK & G \end{bmatrix} \succeq 0, \tag{6.62d}$$

$$\mathbf{W} = \begin{bmatrix} I_n & \Phi^{-1}G\,[K \quad 0]^T \\ G\Phi^{-T} & \mathbf{W}_{22} & L^T \\ [K \quad 0] & L & \mathbf{W}_{33} \end{bmatrix}, \tag{6.62e}$$

$$K \in \mathcal{K}, \tag{6.62f}$$

$$\mathbf{W}_{33} \in \mathcal{K}^2, \tag{6.62g}$$

$$\mathbf{W} \succeq 0, \tag{6.62h}$$

over $K \in \mathbb{R}^{m\times r}$, $P \in \mathbb{S}_n$, $G \in \mathbb{S}_n$, $L \in \mathbb{R}^{m\times n}$, $M \in \mathbb{S}_r$ and $\mathbf{W} \in \mathbb{S}_{2n+m}$.

It should be noted that the constraint (6.62d) ensures that the relation $M = (BK)^T G^{-1}(BK)$ occurs at optimality.

Theorem 6.11. *The problem* (6.62) *is a convex relaxation of the SODC problem. Furthermore, the relaxation is exact if and only if it possesses a solution* $(K^{opt}, L^{opt}, P^{opt}, G^{opt}, M^{opt}, \mathbf{W}^{opt})$ *such that rank*$\{\mathbf{W}^{opt}\} = n$.

Proof 6.14. *Since the proof is similar to that of the infinite-horizon case presented earlier, it is omitted here.*

For the retrieval of a near-optimal controller, the direct recovery method delineated for the infinite-horizon ODC problem can be readily deployed. However, the indirect recovery method requires some modifications, which will be explained below. Let $(K^{\text{opt}}, L^{\text{opt}}, P^{\text{opt}}, G^{\text{opt}}, M^{\text{opt}}, \mathbf{W}^{\text{opt}})$ denote a solution of the computationally-cheap relaxation of SODC. A near-optimal controller K for SODC may be recovered by minimizing

$$\langle K^T(B^T(G^{\text{opt}})^{-1}B + R)K, \Sigma_v\rangle + \alpha\|K\|_F^2 + \varepsilon \times \gamma \tag{6.63a}$$

subject to

$$\begin{bmatrix} (G^{\text{opt}})^{-1} - Q + \gamma I_n & (A+BKC)^T & (KC)^T \\ (A+BKC) & G^{\text{opt}} & 0 \\ (KC) & 0 & R^{-1} \end{bmatrix} \succ 0 \tag{6.63b}$$

$$K \in h_1 N_1 + \ldots + h_l N_l. \tag{6.63c}$$

over $K \in \mathbb{R}^{m\times r}$, $h \in \mathbb{R}^l$ and $\gamma \in \mathbb{R}$, where ε is a pre-specified nonnegative number. This is a convex program.

6.6 Extension to Dynamic Controllers

Consider the problem of finding an optimal fixed-order dynamic controller with a pre-specified structure. To formulate the problem, denote the unknown controller as

$$\begin{cases} z_c[\tau+1] = A_c z_c[\tau] + B_c y[\tau] \\ u[\tau] = C_c z_c[\tau] + D_c y[\tau] \end{cases} \tag{6.64}$$

where $z_c[\tau] \in \mathbb{R}^{n_c}$ represents the state of the controller, and n_c denotes its known degree. The unknown quadruple (A_c, B_c, C_c, D_c) must belong to a given polytope $\mathcal{K}$. More precisely, A_c, B_c, C_c, and D_c are often required to be block matrices with certain forced zero blocks. It is shown in [?] how the design of a fixed-order distributed controller for an interconnected system adopts the above formulation. The augmentation of the system (6.1) with the above unknown controller leads to the closed-loop system $\tilde{x}[\tau+1] = \tilde{A}\tilde{x}[\tau]$, where $\tilde{x}[\tau] = \begin{bmatrix} x[\tau+1]^T \; z_c[\tau+1]^T \end{bmatrix}^T$ and

$$\tilde{A} = \begin{bmatrix} A + BD_cC & BC_c \\ B_cC & A_c \end{bmatrix} \tag{6.65}$$

Note that this closed-loop system reduces to $x[\tau+1] = (A+BKC)x[\tau]$ in the static case. Since $\tilde{A}$ is a linear structured matrix with respect to (A_c, B_c, C_c, D_c), the state evolution equation $\tilde{x}[\tau+1] = \tilde{A}\tilde{x}[\tau]$ is bilinear, similar to its static counterpart $x[\tau+1] = (A+BKC)x[\tau]$. Hence, the parameterized matrix $\tilde{A}$ plays the role of $A+BKC$, which makes it possible to naturally generalize all results of this work to the dynamic case in both finite- and infinite-horizon cases. Note that the existence of a Lyapunov matrix guarantees the stability of $\tilde{A}$ or the internal stability of the system.

6.7 Numerical Examples

In what follows, we offer multiple experiments on random systems and mass-spring systems. More simulations are provided in [38].

6.7.1 Random Systems

Consider the system (6.1) with $n = 5$ and $m = r = 3$. The goal is to design a decentralized static controller $u[\tau] = Ky[\tau]$ (i.e., a diagonal matrix K) minimizing the cost function

$$\left(\sum_{\tau=0}^{20} x[\tau]^T x[\tau] + u[\tau]^T u[\tau] \right) + 10^{-3} \|K\|_F \tag{6.66}$$

This function accounts for the state regulation, input energy, and controller gain. The SDP relaxation problems (6.22), (6.26) and (6.33) have a 235×235, 168×168 and 29×29 matrix variables, respectively. According to Corollary 6.1, it is guaranteed that the sparse SDP relaxation problem (6.22) has a solution W^{opt} with rank at most 3 (i.e., at least 233 eigenvalues of this

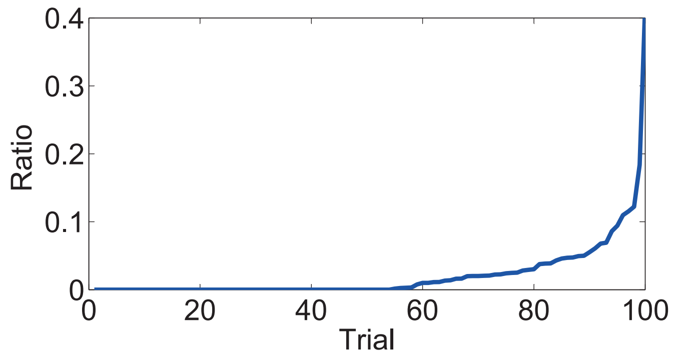
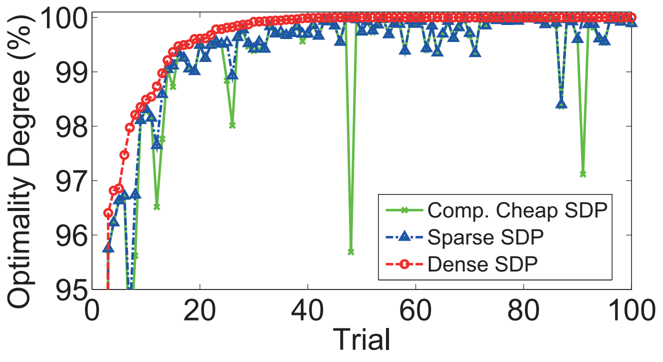


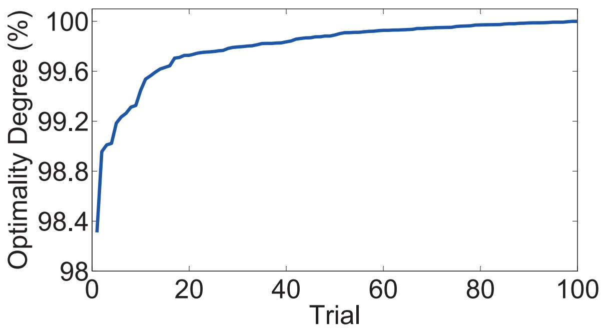
Figure 6.5: The ratio $\frac{\lambda_2}{\lambda_1}$ obtained from the dense SDP relaxation of the finite-horizon ODC Problem (6.26) for 100 random systems.

solution must be zero), independent of the values of the matrices A, B, C, and $x[0]$. Note that this result does not imply that all solutions of problem (6.22) have rank at most 3, but Theorem 6.1 can be used to find such a low-rank solution.

Since real-world systems are normally highly structured in many ways, we consider some structure for the system under study by assuming that B can be expressed as $[b \quad b \quad b]$ for some vector $b \in \mathbb{R}^5$. Assume that A, b, and $x[0]$ are normal random variables with the standard deviations 0.2, 1, and 1, respectively, while C is equal to $[I_3 \quad 0_{3\times 2}]$. We generated 100 random systems according to the above probability distributions for the parameters of the system and checked the rank of a low-rank solution of the sparse, dense, and computationally-cheap SDP relaxation problems for every trial. Let λ_1 and λ_2 denote the largest and the second largest eigenvalues of W^{opt} associated with the dense relaxation. We arranged the obtained 100 ratios $\frac{\lambda_2}{\lambda_1}$ in ascending order and subsequently labeled their corresponding trials as $1, 2, \ldots, 100$. Figure 6.5 plots the ratio $\frac{\lambda_2}{\lambda_1}$ for the ordered trials. It can be observed that this ratio is equal to 0 for 53 trials, implying that the dense SDP relaxation has found the solution of the ODC problem for 53 samples of the system. In addition, $\frac{\lambda_2}{\lambda_1}$ is less than 0.1 in 95 trials. Also, three near-global solutions of the ODC problem were found using different relaxations in all 100 cases. Figure 6.6 (a) depicts the (global) optimality degrees of these solutions after re-arranging the trials based on their associated optimality degrees for the dense SDP relaxation problem. Optimality degree



(a)



(b)

Figure 6.6: Optimal degrees of different relaxations for 100 random systems.

is defined as

$$\text{Optimality degree (\%)} = 100 - \frac{\text{upper bound - lower bound}}{\text{upper bound}} \times 100$$

where "upper bound" and 'lower bound" denote the cost of the near-global controller recovered using the direct method and the optimal SDP cost, respectively. The optimality degree is an upper bound on the closeness of the cost of the near-optimal controller to the minimum cost, which is expressed in percentage. Notice that the employed optimality measure evaluates the global performance within the specified set of controllers. For example, the optimality degree of 100% means that a globally optimal controller is found among all <u>linear static</u> structured controllers.

As an alternative, we solved a penalized SDP relaxation with the penalty term $\Psi(W) = 0.5\ \text{trace}\{W\}$ added to the objective of the SDP relaxation. Interestingly, the matrix $\tilde{W}^{\text{opt}}$

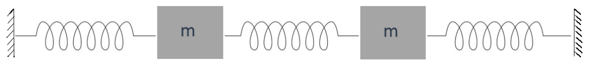


Figure 6.7: Mass-spring system with two masses

became rank 1 for all of the 100 trials. Figure 6.6 (b) depicts the optimality degrees associated with the penalized dense SDP relaxation problem of the 100 random systems. It can be seen that the optimality degree is greater than 99.8% for 69 trials and is never less than 98.2%.

6.7.2 Mass-Spring Systems

In this subsection, the aim is to evaluate the performance of the developed controller design techniques in Lyapunov domain on the *Mass-Spring* system, as a classical physical system. Consider a mass-spring system consisting of N masses. This system is exemplified in Figure 6.7 for $N = 2$. The system can be modeled in the continuous-time domain as

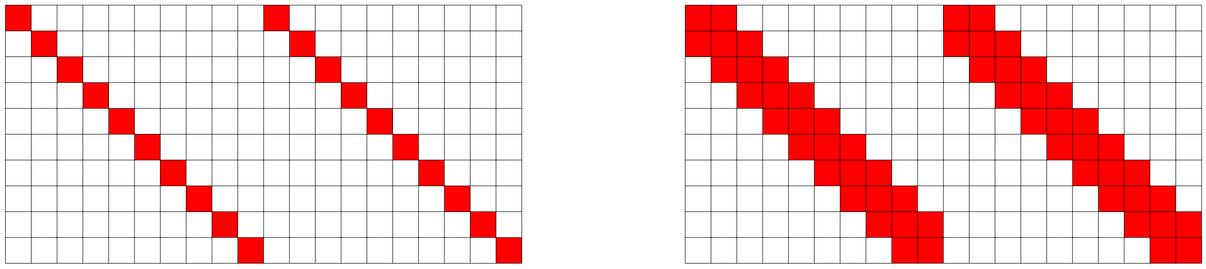
$$\dot{x}_c(t) = A_c x_c(t) + B_c u_c(t) \tag{6.67}$$

where the state vector $x_c(t)$ can be partitioned as $[o_1(t)^T \ o_2(t)^T]$ with $o_1(t) \in \mathbb{R}^n$ equal to the vector of positions and $o_2(t) \in \mathbb{R}^n$ equal to the vector of velocities of the N masses. We assume that $N = 10$ and adopt the values of A_c and B_c from [80]. The goal is to design a static sampled-data controller with a pre-specified structure (i.e., the controller is composed of a sampler, a static discrete-time structured controller and a zero-order holder). Consider two different control structures shown in Figure 6.8. The free parameters of each controller are colored in red in this figure. Notice that Structure (a) corresponds to a fully decentralized controller, where each local controller has access to the position and velocity of its associated mass. In contrast, Structure (b) allows limited communications between neighboring local controllers. Two ODC problems will be solved for these structures below.

Infinite-Horizon Deterministic ODC: In this experiment, we first discretize the system with the sampling time of 0.1 second and denote the obtained system as

$$x[\tau+1] = Ax[\tau] + Bu[\tau], \qquad \tau = 0, 1, \ldots \tag{6.68}$$

It is aimed to design a constrained controller $u[\tau] = Kx[\tau]$ to minimize the cost function $\sum_{\tau=0}^{\infty} \left(x[\tau]^T x[\tau] + u[\tau]^T u[\tau] \right)$. Consider 100 randomly-generated initial states $x[0]$ with entries



(a) Decentralized (b) Distributed

Figure 6.8: Two different structures (decentralized and distributed) for the controller K: the free parameters are colored in red (uncolored entries are set to zero).

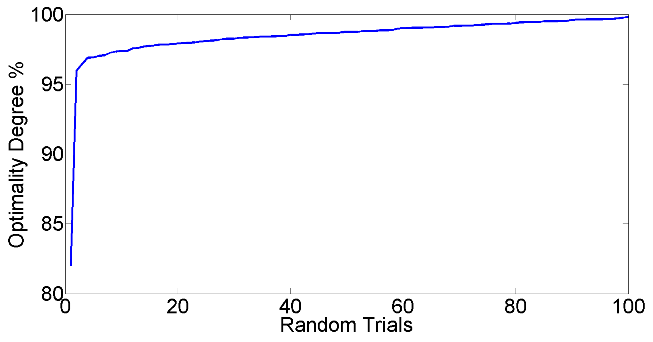
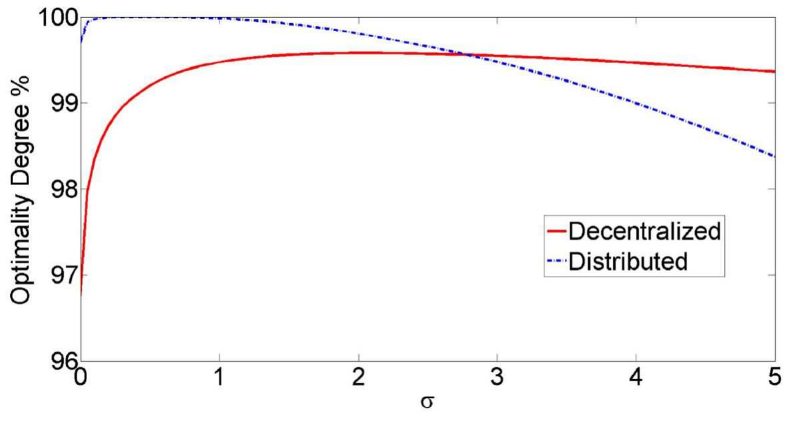


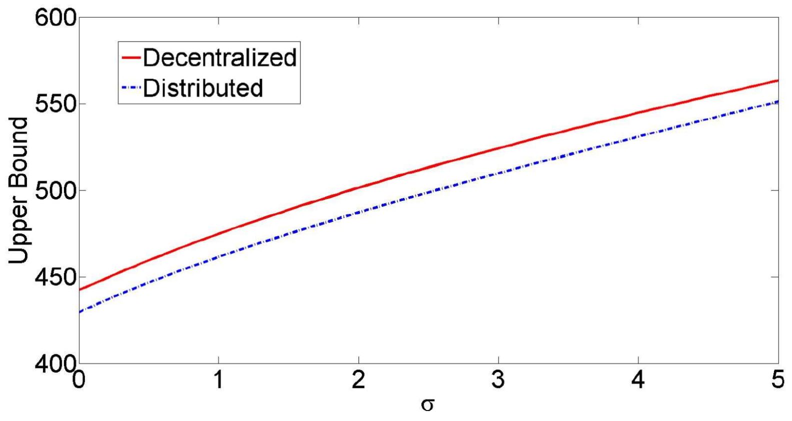
Figure 6.9: Optimality degree (%) of the decentralized controller $\hat{K}$ for a mass-spring system under 100 random initial states.

drawn from a normal distribution. We solved the computationally-cheap SDP relaxation of the infinite-horizon ODC problem combined with the direct recovery method to design a controller of Structure (a) minimizing the above cost function. The optimality degrees of the controllers designed for these 100 random trials are depicted in Figure 6.9. As can be seen, the optimality degree is better than 95% for more than 98 trials. It should be mentioned that all of these controllers stabilize the system.

Infinite-Horizon Stochastic ODC: Assume that the system is subject to both input disturbance and measurement noise. Consider the case $\Sigma_d = I_n$ and $\Sigma_v = \sigma I_r$, where σ varies from 0 to 5. Using the computationally-cheap relaxation problem (6.62) in conjunction with the indirect recovery method, a near-optimal controller is designed for each of the aforementioned control



(a) Optimality degree



(b) Cost of near-optimal controller

Figure 6.10: The optimality degree and optimal cost of the near-optimal controller designed for the mass-spring system for two different control structures. The noise covariance matrix Σ_v is assumed to be equal to σI_r, where σ varies over a wide range.

structures under various noise levels. The results are reported in Figure 6.10. The designed structured controllers are all stable with optimality degrees higher than 95% in the worst case and close to 99% in many cases.

6.8 Conclusions

This chapter studies the optimal distributed control (ODC) problem for discrete-time deterministic and stochastic systems. The objective is to design a fixed-order distributed controller with a pre-determined structure to minimize a quadratic cost functional. Both time domain and Lyapunov domain formulations of the ODC problem are cast as rank-constrained optimization problems with only one non-convex constraint requiring the rank of a variable matrix to be 1. We propose semidefinite programming (SDP) relaxations of these problems. The notion of tree decomposition is exploited to prove the existence of a low-rank solution for the SDP relaxation problems with rank at most 3. This result can be a basis for a better understanding of the complexity of the ODC problem because it states that almost all eigenvalues of the SDP solution are zero. Moreover, multiple recovery methods are proposed to round the rank-3 solution to rank 1, from which a near-global controller may be retrieved. Computationally-cheap relaxations are also developed for finite-horizon, infinite-horizon, and stochastic ODC problems. These relaxations are guaranteed to exactly solve the LQR and H_2 problems for the classical centralized control problem. The results are tested on multiple examples.

Chapter 7

Conclusion

In this dissertation, I presented my work on probabilistic modeling and optimization for scalable inference. Probabilistic models are powerful techniques for expressing underlying structures in data and inferring hidden information from them. Training probabilistic models is challenging in that it mainly requires dealing with non-convex optimization problems. In this dissertation, I presented my research results on designing interpretable models that could capture latent themes from data, deriving learning algorithms to train the model, and relaxing hard optimization problems.

I presented a new Bayesian nonparametric model called *beta process subspace analysis* (BPSA) for dictionary learning that sparsely codes signals in latent subspaces in Chapter 3. The is model an extension of related methods such as Beta Process Factor Analysis (BPFA) and Mixture of Factor Analysis (MFA). Using beta and gamma processes, it can infer both the number of subspaces and the dimensionality of each subspace. I derived a new Maximizing a Posteriori and Expectation Maximization (MAP-EM) based algorithm that is related to variational inference and the Orthogonal Matching Pursuit (OMP) algorithm. I illustrated the model procedure on Tiny Images data set and demonstrated the advantage of sparse coding with subspaces on denoising problems.

The proposed learning algorithm is further investigated in Chapter 2. I proposed probabilistic orthogonal matching pursuit (PrOMP) for sparse data representation. Our probabilistic approach extends OMP, making it suitable for statistical dictionary learning models with Bayesian nonparametric priors. I derived theory for PrOMP similar to that of OMP, and discussed how PrOMP can improve existing dictionary learning models. We evaluated the performance on image denoising and compresses sensing for magnetic resonance imaging (CS-MRI), showing that PrOMP for BPFA improves the classic K Singular Value Decomposition (K-SVD) model, as well as Markov Chain Monte Carlo (MCMC) sampling for the same Bayesian nonparametric prior dictionary learning model.

Next, I investigated model developments for sequential data in Chapter 4. I presented a mixed membership recurrent neural network (MM-RNN) approach for modeling multiple sequences. The model was motivated by the observation that, in many sequential data sets the sequential information is not of the same value across the sequence. As more time passes between observations, the distribution on the next observation may be better modeled as independent from some initial group-specific distribution. To this end, we made a simple modification to the RNN architecture by generating a unique base vector for each group and use a weighted combination of this base vector with the RNN hidden state to make predictions. The weight emphasizes the RNN in the part of the sequences that is densely sampled, and emphasizes the group-specific i.i.d. model when two consecutive observations are spread far apart in time. I demonstrated on two online shopping data sets that this combination of sequential/non-sequential modeling can allow for the RNN to focus on learning to make better predictions when sequential information is meaningful, and to defer to the base model when much time has passed in a smooth transition.

In Chapter 5, I derived convex relaxation for variational inference (CRVI), a method to learn parameters of approximate posterior distributions using mean-field variational inference. I focused on Bayesian linear regression and sparse coding models. By lifting the domain of the optimization, we were able to relax the non-convex parts of the variational objective function and approximate the variational parameters. Graph theoretic tools enabled us to quantify the exactness of this approximation, and estimate the closeness of the obtained solution to the global optimal solution. I showed that CRVI can significantly improve the traditional coordinate ascent

(CAVI) optimization technique on various datasets for sparse Bayesian linear regression and sparse coding for nonparametric factor analysis.

In the last Chapter, 6, I studied the optimal distributed control (ODC) problem for discrete-time deterministic and stochastic systems. The objective was to design a fixed-order distributed controller with a pre-determined structure to minimize a quadratic cost functional. Both time domain and Lyapunov domain formulations of the ODC problem were cast as rank-constrained optimization problems with only one non-convex constraint requiring the rank of a variable matrix to be 1. We proposed semidefinite programming (SDP) relaxations of these problems. The notion of tree decomposition was exploited to prove the existence of a low-rank solution for the SDP relaxation problems with rank at most 3. This result can be a basis for a better understanding of the complexity of the ODC problem because it states that almost all eigenvalues of the SDP solution are zero. Moreover, multiple recovery methods were proposed to round the rank-3 solution to rank 1, from which a near-global controller may be retrieved. Computationally-cheap relaxations are also developed for finite-horizon, infinite-horizon, and stochastic ODC problems. These relaxations are guaranteed to exactly solve the LQR and H_2 problems for the classical centralized control problem. The results are tested on multiple examples.

The ideas explored in theses can lead to various future research directions. Bayesian nonparametric models and PrOMP are powerful techniques that can be further extended for applications in online datasets. This will make the inference process fast, and scalable to larger datasets. This could lead to a generalized way of designing mixed membership recurrent neural networks for sequential datasets. The convex relaxation technique proposed in this dissertation can be investigated for becoming more suitable to larger models and datasets. The semidefinite constraint is a bottleneck in the computation. Developing a technique to break down this constraint into smaller portions or replacing it with other conditions can significantly make the process faster.

Bibliography

[1] M. Aharon, M. Elad, and A. Bruckstein. K-SVD: An algorithm for designing overcomplete dictionaries for sparse representation. *IEEE Transactions on signal processing*, 54(11):4311–4322, 2006.

[2] M. Aharon, M. Elad, A. Bruckstein, and Y. Katz. K-SVD: An algorithm for denoising of overcomplete dictionaries for sparse representation. *IEEE Transaction on Signal Processing*, 54, 2006.

[3] E. M. Airoldi, D. Blei, E. A. Erosheva, and S. E. E. Fienberg. *Handbook of mixed membership models and their applications*. CRC Press, 2014.

[4] N. Akhtar, A. Mian, and F. Porikli. Joint discriminative Bayesian dictionary and classifier learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[5] B. Bamieh, F. Paganini, and M. A. Dahleh. Distributed control of spatially invariant systems. *IEEE Transactions on Automatic Control*, 47(7):1091–1107, 2002.

[6] B. Bamieh and P. G. Voulgaris. A convex characterization of distributed control problems in spatially invariant systems with communication constraints. *Systems & Control Letters*, 54(6):575 – 583, 2005.

[7] Y. Bengio and F. Gingras. Recurrent neural networks for missing or asynchronous data. In *Advances in neural information processing systems*, pages 395–401, 1996.

[8] E. R. Berlekamp. Factoring polynomials over large finite fields. *Mathematics of Computation*, 1970.

[9] D. Bienstock and G. Munoz. Lp formulations for mixed-integer polynomial optimization problems. 2015.

[10] C. Bishop. Pattern recognition and machine learning. *Springer*, 2006.

[11] D. M. Blei and J. D. Lafferty. Dynamic topic models. In *Proceedings of the 23rd international conference on Machine learning*, pages 113–120. ACM, 2006.

[12] D. M. Blei, J. D. Lafferty, et al. A correlated topic model of science. *The Annals of Applied Statistics*, 1(1):17–35, 2007.

[13] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. *Journal of machine Learning research*, 3(Jan):993–1022, 2003.

[14] H. L. Bodlaender. A tourist guide through treewidth. *Acta cybernetica*, 11(1-2):1, 1994.

[15] F. Borrelli and T. Keviczky. Distributed LQR design for identical dynamically decoupled systems. *IEEE Transactions on Automatic Control*, 53(8):1901–1912, 2008.

[16] L. Bottou. Online learning and stochastic approximation. *Online Learning in Neural Networks*, 1998.

[17] S. Boyd and L. Vandenberghe. Convex optimization. *Cambridge University Press*, 2004.

[18] S. Boyd and L. Vandenberghe. *Convex Optimization.* Cambridge, 2004.

[19] O. Cappé and E. Moulines. On-line expectation–maximization algorithm for latent data models. *Journal of the Royal Statistical Society: Series B*, 71(3):593–613, 2009.

[20] D. Carlson, E. Haynsworth, and T. Markham. A generalization of the schur complement by means of the moore-penrose inverse. *SIAM Journal on Applied Mathematics*, 26(1):169–175, 1974.

[21] Z. Che, S. Purushotham, K. Cho, D. Sontag, and Y. Liu. Recurrent neural networks for multivariate time series with missing values. *Scientific reports*, 8(1):6085, 2018.

[22] M. Chen, J. Silva, J. Paisley, C. Wang, D. Dunson, and L. Carin. Compressive sensing on manifolds using a nonparametric mixture of factor analyzers: Algorithm and performance bounds. *IEEE Transactions on Signal Processing*, 58(12):6140–6155, 2010.

[23] S. S. Chen, D. L. Donoho, and M. A. Saunders. Atomic decomposition by basis pursuit. *SIAM review*, 43(1):129–159, 2001.

[24] Y. Chen, X. Ye, and F. Huang. A novel method and fast algorithm for mr image reconstruction with significantly under-sampled data. *Inverse Problems and Imaging*, 4(2):223–240, 2010.

[25] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

[26] E. Choi, M. T. Bahadori, A. Schuetz, W. F. Stewart, and J. Sun. Doctor ai: Predicting clinical events via recurrent neural networks. In *Machine Learning for Healthcare Conference*, pages 301–318, 2016.

[27] R. D'Andrea and G. Dullerud. Distributed control design for spatially interconnected systems. *IEEE Transactions on Automatic Control*, 48(9):1478–1495, 2003.

[28] R. A. Date and J. H. Chow. A parametrization approach to optimal H_2 and H_∞ decentralized control problems. *Automatica*, 29(2):457 – 463, 1993.

[29] G. A. de Castro and F. Paganini. Convex synthesis of localized controllers for spatially invariant systems. *Automatica*, 38(3):445 – 456, 2002.

[30] A. B. Dieng, C. Wang, J. Gao, and J. Paisley. TopicRNN: A recurrent neural network with long-range semantic dependency. In *International Conference on Learning Representations*, 2016.

[31] D. L. Donoho, Y. Tsaig, I. Drori, and J.-L. Starck. Sparse solution of underdetermined systems of linear equations by stagewise orthogonal matching pursuit. *IEEE Transactions on Information Theory*, 58(2):1094–1121, 2012.

[32] N. Du, H. Dai, R. Trivedi, U. Upadhyay, M. Gomez-Rodriguez, and L. Song. Recurrent marked temporal point processes: Embedding event history to vector. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1555–1564. ACM, 2016.

[33] G. Dullerud and R. D'Andrea. Distributed control of heterogeneous systems. *IEEE Transactions on Automatic Control*, 49(12):2113–2128, 2004.

[34] K. Dvijotham, E. Theodorou, E. Todorov, and M. Fazel. Convexity of optimal linear controller design. *Conference on Decision and Control*, 2013.

[35] B. Efron, T. Hastie, I. Johnstone, R. Tibshirani, et al. Least angle regression. *The Annals of statistics*, 32(2):407–499, 2004.

[36] J. L. Elman. Finding structure in time. *Cognitive science*, 14(2):179–211, 1990.

[37] M. Fardad, F. Lin, and M. R. Jovanovic. On the optimal design of structured feedback gains for interconnected systems. In *48th IEEE Conference on Decision and Control*, pages 978–983, 2009.

[38] G. Fazelnia, R. Madani, A. Kalbat, and J. Lavaei. Convex relaxation for distributed frequency control in power systems. 2015.

[39] G. Fazelnia, R. Madani, A. Kalbat, and J. Lavaei. Convex relaxation for optimal distributed control problems. *IEEE Transactions on Automatic Control*, 62(1):206–221, 2016.

[40] G. Fazelnia, R. Madani, A. Kalbat, and J. Lavaei. Convex relaxation for optimal distributed control problems. *IEEE Transactions on Automatic Control*, 62(1):206–221, 2017.

[41] G. Fazelnia and J. Paisley. Bayesian nonparametric latent subspace analysis. *Preprint http://www.columbia.edu/∼gf2293/FazelniaBPSA.pdf.*

[42] G. Fazelnia and J. Paisley. Probabilistic orthogonal matching pursuit. *Preprint http://www.columbia.edu/∼gf2293/FazelniaPROMP.pdf.*

[43] G. Fazelnia and J. Paisley. Crvi: Convex relaxation for variational inference. In *International Conference on Machine Learning*, pages 1476–1484, 2018.

[44] M. Fukuda, M. Kojima, K. Murota, and K. Nakata. Exploiting sparsity in semidefinite programming via matrix completion I: general framework. *SIAM J. Optimization*, 2000.

[45] C. M. K. W. G. Fazelnia, M. Ibrahim and J. Paisley. Mixed membership recurrent neural networks. *Preprint https://arxiv.org/abs/1812.09645.*

[46] A. Gelfand and A. Smith. Sampling based approaches to calculating marginal densities. *Journal of the American Statistical Association*, 1990.

[47] J. Geromel, J. Bernussou, and P. Peres. Decentralized control through parameter space optimization. *Automatica*, 30(10):1565 – 1578, 1994.

[48] Z. Ghahramani. Probabilistic machine learning and artificial intelligence. *Nature*, 521(7553):452, 2015.

[49] Z. Ghahramani and G. Hinton. The EM algorithm for mixtures of factor analyzers. *Technical report, Department of Computer Science, University of Toronto*, 1996.

[50] M. X. Goemans and D. P. Williamson. Improved approximation algorithms for maximum cut and satisability problems using semidefinite programming. *Journal of the ACM*, 42:1115–1145, 1995.

[51] M. X. Goemans and D. P. Williamson. Approximation algorithms for max-3-cut and other problems via complex semidefinite programming. *Journal of Computer and System Sciences*, 68:422–470, 2004.

[52] T. Goldstein and S. Osher. The split Bregman method for L1-regulized problems. *SIAM Journal on Imaging Sciences*, 2(2):223–243, 2009.

[53] M. Grant and S. Boyd. Graph implementations for nonsmooth convex programs. In V. Blondel, S. Boyd, and H. Kimura, editors, *Recent Advances in Learning and Control*, Lecture Notes in Control and Information Sciences, pages 95–110. Springer-Verlag Limited, 2008.

[54] M. Grant and S. Boyd. CVX: Matlab software for disciplined convex programming, version 2.1. `http://cvxr.com/cvx`, 2014.

[55] T. Griffiths and Z. Ghahramani. The Indian buffet process: An introduction and review. *Journal of Machine Learning Research*, 12:1185–1224, 2011.

[56] Y. Guo and D. Schuurmans. Convex relaxations of latent variable training. *Advances in Neural Information Processing Systems*, 2008.

[57] T. Hastie, R. Tibshirani, and M. Wainwright. *Statistical learning with sparsity: the lasso and generalizations*. CRC press, 2015.

[58] W. Hastings. Monte carlo sampling methods using markov chains and their applications. *Biometrika*, 1970.

[59] S. He, Z. Li, and S. Zhang. Approximation algorithms for homogeneous polynomial optimization with quadratic constraints. *Mathematical Programming*, 125:353–383, 2010.

[60] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[61] M. Hoffman, D. Blei, C. Wang, and J. Paisley. Stochastic variational inference. *Journal of Machine Learning*, 14, 2013.

[62] M. D. Hoffman, D. M. Blei, C. Wang, and J. Paisley. Stochastic variational inference. *The Journal of Machine Learning Research*, 14(1):1303–1347, 2013.

[63] W. Huang, F. Sun, L. Cao, D. Zhao, H. Liu, and M. Harandi. Sparse coding and dictionary learning with linear dynamical systems. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3938–3947, 2016.

[64] Y. Huang, J. Paisley, Q. Lin, X. Ding, X. Fu, and X.-P. Zhang. Bayesian nonparametric dictionary learning for compressed sensing mri. *IEEE Transactions on Image Processing*, 23(12):5007–5019, 2014.

[65] A. Hyvarinen and P. Hoyer. Emergence of phase-and shift-invariant features by decomposition of natural images into independent feature subspaces. *Neural computation*, 2000.

[66] H. Jing and A. J. Smola. Neural survival recommender. In *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining*, pages 515–524. ACM, 2017.

[67] M. I. Jordan, Z. Ghahramani, T. S. Jaakkola, and L. K. Saul. An introduction to variational methods for graphical models. *Machine learning*, 37(2):183–233, 1999.

[68] A. Kalbat and J. Lavaei. A fast distributed algorithm for sparse semidefinite programs. 2016.

[69] T. Keviczky, F. Borrelli, and G. J. Balas. Decentralized receding horizon control for large scale dynamically decoupled systems. *Automatica*, 42(12):2105–2115, 2006.

[70] D. Knowles and Z. Ghahramani. Infinite sparse factor analysis and infinite independent components analysis. In *International Conference on Independent Component Analysis and Signal Separation*, pages 381–388. Springer, 2007.

[71] D. Knowles and Z. Ghahramani. Nonparametric Bayesian sparse factor models with application to gene expression modeling. *Annals of Applied Statistics*, 2011.

[72] A. Lamperski and J. C. Doyle. Output feedback H_2 model matching for decentralized systems with delays. *American Control Conference*, 2013.

[73] C. Langbort, R. Chandra, and R. D'Andrea. Distributed control design for systems interconnected over an arbitrary graph. *IEEE Transactions on Automatic Control*, 49(9):1502–1519, 2004.

[74] J. Lavaei. Decentralized implementation of centralized controllers for interconnected systems. *IEEE Transactions on Automatic Control*, 57(7):1860–1865, 2012.

[75] J. Lavaei and A. G. Aghdam. Control of continuous-time LTI systems by means of structurally constrained controllers. *Automatica*, 44(1), 2008.

[76] J. Lavaei and S. H. Low. Zero duality gap in optimal power flow problem. *IEEE Transactions on Power Systems*, 27(1):92–107, 2012.

[77] Q. Le, W. Zou, S. Yeung, and A. Ng. Learning hierarchical invariant spatio-temporal features for action recognition with independent subspace analysis. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2011.

[78] D. D. Lee and H. S. Seung. Algorithms for non-negative matrix factorization. In *Advances in neural information processing systems*, pages 556–562, 2001.

[79] L. Lessard and S. Lall. Optimal controller synthesis for the decentralized two-player problem with output feedback. *American Control Conference*, 2012.

[80] F. Lin, M. Fardad, and M. R. Jovanovi. Design of optimal sparse feedback gains via the alternating direction method of multipliers. *IEEE Transactions on Automatic Control*, 58(9), 2013.

[81] F. Lin, M. Fardad, and M. R. Jovanovic. Augmented lagrangian approach to design of structured optimal state feedback gains. *IEEE Transactions on Automatic Control*, 56(12):2923–2929, 2011.

[82] Z. C. Lipton, D. Kale, and R. Wetzel. Directly modeling missing data in sequences with rnns: Improved classification of clinical time series. In *Machine Learning for Healthcare Conference*, pages 253–270, 2016.

[83] X. Liu, M. Tanaka, and M. Okutomi. Single-image noise level estimation for blind-denoising. *IEEE Transaction on Image Processing*, 22(12):5226–5237, 2004.

[84] M. Lustig, D. L. Donoho, J. M. Santos, and J. M. Pauly. Compressed sensing mri. *IEEE signal processing magazine*, 25(2):72–82, 2008.

[85] R. Madani, G. Fazelnia, S. Sojoudi, and J. Lavaei. Low-rank solutions of matrix inequalities with applications to polynomial optimization and matrix completion problems. *Conference on Decision and Control*, 2014.

[86] R. Madani, G. Fazelnia, S. Sojoudi, and J. Lavaei. Finding low-rank solutions of sparse linear matrix inequalities using convex optimization. *SIAM Journal on Optimization*, 2017.

[87] J. Mairal, F. bach, J. Ponce, and G. Sapiro. Online learning for matrix factorization and sparse coding. *Journal of Machine Learning*, 11, 2010.

[88] S. Mallat and Z. Zhang. Matching pursuit with time-frequency dictionaries. Technical report, Courant Institute of Mathematical Sciences New York United States, 1993.

[89] N. Matni and J. C. Doyle. A dual problem in H_2 decentralized control subject to delays. *American Control Conference*, 2013.

[90] N. Motee and A. Jadbabaie. Optimal control of spatially distributed systems. *IEEE Transactions on Automatic Control*, 53(7):1616–1629, 2008.

[91] D. Needell and J. A. Tropp. Cosamp: Iterative signal recovery from incomplete and inaccurate samples. *Applied and computational harmonic analysis*, 26(3):301–321, 2009.

[92] D. Needell and R. Vershynin. Uniform uncertainty principle and signal recovery via regularized orthogonal matching pursuit. *Foundations of computational mathematics*, 9(3):317–334, 2009.

[93] Y. Nesterov. Semidefinite relaxation and nonconvex quadratic optimization. *Optimization Methods and Software*, 9:141–160, 1998.

[94] H. Nickisch and M. W. Seeger. Convex variational bayesian inference for large scale generalized linear models. *Proceedings of the 26th Annual International Conference on Machine Learning*, 2009.

[95] J. Paisley and L. Carin. Nonparametric factor analysis with beta process priors. *International Conference on Machine Learning*, 2009.

[96] J. Paisley and L. Carin. Nonparametric factor analysis with beta process priors. In *International Conference on Machine Learning*, 2009.

[97] J. Paisley and M. I. Jordan. A constructive definition of the beta process. *arXiv preprint, arXiv: 1604.0068*, 2016.

[98] P. Parrilo. Semidefinite programming relaxations for semialgebraic problems. *Mathematical Programming*, 2003.

[99] S. Parveen and P. Green. Speech recognition with missing data using recurrent neural nets. In *Advances in Neural Information Processing Systems*, pages 1189–1195, 2002.

[100] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer. Automatic differentiation in pytorch. In *NIPS-W*, 2017.

[101] G. Pataki. On the rank of extreme matrices in semidefinite programs and the multiplicity of optimal eigenvalues. *Mathematics of Operations Research*, 23, 1998.

[102] Y. C. Pati, R. Rezaiifar, and P. S. Krishnaprasad. Orthogonal matching pursuit: Recursive function approximation with applications to wavelet decomposition. In *Signals, Systems and Computers, 1993. 1993 Conference Record of The Twenty-Seventh Asilomar Conference on*, pages 40–44. IEEE, 1993.

[103] T. Pham, T. Tran, D. Phung, and S. Venkatesh. Deepcare: A deep dynamic memory model for predictive medicine. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 30–41. Springer, 2016.

[104] X. Qi, M. Salapaka, P. Voulgaris, and M. Khammash. Structured optimal and robust control with multiple criteria: a convex solution. *IEEE Transactions on Automatic Control*, 49(10):1623–1640, 2004.

[105] Y. Quan, Y. Xu, Y. Sun, Y. Huang, and H. Ji. Sparse coding for classification via discrimination ensemble. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[106] A. Rantzer. Distributed control of positive systems. *http://arxiv.org/abs/1203.0047*, 2012.

[107] S. Ravishankar and Y. Bresler. Mr image reconstruction from highly undersampled k-space data by dictionary learning. *IEEE transactions on medical imaging*, 30(5):1028–1041, 2011.

[108] B. Recht, M. Fazel, and P. A. Parrilo. Guaranteed minimum-rank solutions of linear matrix equations via nuclear norm minimization. *SIAM review*, 52(3):471–501, 2010.

[109] M. Rotkowitz and S. Lall. A characterization of convex problems in decentralized control. *IEEE Transactions on Automatic Control*, 51(2):274–286, 2006.

[110] M. Rotkowitz and N. Martins. On the nearest quadratically invariant information constraint. *IEEE Transactions on Automatic Control*, 57(5):1314–1319, 2012.

[111] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533, 1986.

[112] G. Scorletti and G. Duc. An LMI approach to dencentralized H_∞ control. *International Journal of Control*, 74(3):211–224, 2001.

[113] M. W. Seeger and H. Nickisch. Large scale bayesian inference and experimental design for sparse linear models. *SIAM Journal on Imaging Sciences*, 4(1):166–199, 2011.

[114] S. Sertoglu and J. Paisley. Scalable Bayesian nonparametric dictionary learning. *European Signal Processing Conference*, 2015.

[115] A. Shah, D. Knowles, and Z. Ghahramani. An empirical study of stochastic variational algorithms for the beta bernoulli process. *International Conference on Machine Learning*, 2015.

[116] P. Shah and P. Parrilo. H_2-optimal decentralized control over posets: A state-space solution for state-feedback. *IEEE Transactions on Automatic Control,*, 58(12):3084–3096, Dec 2013.

[117] D. D. Siljak. Decentralized control and computations: status and prospects. *Annual Reviews in Control*, 20:131–141, 1996.

[118] S. Sojoudi and J. Lavaei. Physics of power networks makes hard optimization problems easy to solve. *IEEE Power & Energy Society General Meeting*, 2012.

[119] S. Sojoudi and J. Lavaei. On the exactness of semidefinite relaxation for nonlinear optimization over graphs: Part I. *IEEE Conference on Decision and Control*, 2013.

[120] S. Sojoudi and J. Lavaei. On the exactness of semidefinite relaxation for nonlinear optimization over graphs: Part II. *IEEE Conference on Decision and Control*, 2013.

[121] S. Sojoudi and J. Lavaei. Exactness of semidefinite relaxations for nonlinear optimization problems with underlying graph structure. to appear in *SIAM Journal on Optimization*, 2014.

[122] J. F. Sturm and S. Zhang. On cones of nonnegative quadratic functions. *Mathematics of Operations Research*, 28, 2003.

[123] T. Tanaka and C. Langbort. The bounded real lemma for internally positive systems and H-infinity structured static state feedback. *IEEE Transactions on Automatic Control*, 56(9):2218–2223, 2011.

[124] R. Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 267–288, 1996.

[125] R. Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society B*, 1996.

[126] A. Torralba, R. Fergus, and W. T. Freeman. 80 million tiny images: A large data set for nonparametric object and scene recognition. *IEEE transactions on pattern analysis and machine intelligence*, 30(11):1958–1970, 2008.

[127] V. Tresp and T. Briegel. A solution for missing data in recurrent neural networks with an application to blood glucose prediction. In *Advances in Neural Information Processing Systems*, pages 971–977, 1998.

[128] J. A. Tropp. Greed is good: Algorithmic results for sparse approximation. *IEEE Transactions on Information theory*, 50(10):2231–2242, 2004.

[129] J. N. Tsitsiklis and M. Athans. On the complexity of decentralized decision making and detection problems. *Conference on Decision and Control*, 1984.

[130] L. Vandenberghe and S. Boyd. Semidefinite programming. *SIAM Review*, 1996.

[131] L. Vandenberghe and S. Boyd. Semidefinite programming. *SIAM Review*, 1996.

[132] B. Vassøy, M. Ruocco, E. de Souza da Silva, and E. Aune. Time is of the essence: a joint hierarchical rnn and point process model for time and item predictions. In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*, pages 591–599. ACM, 2019.

[133] M. J. Wainwright and M. I. Jordan. Graphical models, exponential families, and variational inference. *Foundations and Trends in Machine Learning*, 2008.

[134] Z. Wang, A. Bovik, H. Sheikh, and E. Simoncelli. Image quality assessment: From error visibility to structural similarity. *IEEE Transaction on Image Processing*, 13(4):600–612, 2004.

[135] H. S. Witsenhausen. A counterexample in stochastic optimum control. *SIAM Journal of Control*, 6(1), 1968.

[136] M. Yang, D. Dai, L. Shen, and L. Van Gool. Latent dictionary learning for sparse representation based classification. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2014.

[137] J. S. Yedidia, W. T. Freeman, and Y. Weiss. Bethe free energy, kikuchi approximations, and belief propagation algorithms. *IEEE Transactions on Information Theory*, 51(7):2282–2312, 2005.

[138] M. Zaheer, A. Ahmed, and A. J. Smola. Latent lstm allocation joint clustering and non-linear dynamic modeling of sequential data. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 3967–3976. JMLR. org, 2017.

[139] G. Zhai, M. Ikeda, and Y. Fujisaki. Decentralized H_∞ controller design: a matrix inequality approach using a homotopy method. *Automatica*, 37(4):565 – 572, 2001.

[140] M. Zhou, H. Chen, J. Paisley, L. Ren, L. Li, Z. Xing, D. Dunson, G. Sapiro, and L. Carin. Nonparametric Bayesian dictionary learning for noisy and incomplete images. *IEEE Transaction on Image Processing*, 21(1):130–144, 2012.