The Dissertation Committee for Oscar Hernan Madrid Padilla certifies that this is the approved version of the following dissertation:

Constrained estimation via the fused lasso and some generalizations

Committee:

James G. Scott , Supervisor

Constantine Caramanis

Purnamrita Sarkar

Mingyuan Zhou

Constrained estimation via the fused lasso and some generalizations

by

Oscar Hernan Madrid Padilla, B.S.

DISSERTATION

Presented to the Faculty of the Graduate School of

The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

DOCTOR OF PHILOSOPHY

THE UNIVERSITY OF TEXAS AT AUSTIN

May 2017

Dedicated to my parents.

Acknowledgments

I would like to thank James Scott, who has been a very supportive advisor. He has been patient, helpful, and has made me feel free about my research. For all this I will always be thankful to him.

I would also like to thank Mingyuan Zhou for his guidance during the early years of my stay in Austin. It was him who encouraged me to start doing research very early on in the PhD program.

I would like to specially acknowledge James Sharpnack fo being such supportive collaborator. I had many stimulating conversations with him that have made me a much better researcher.

I am also very grateful to have worked along side great researchers: Nick Polson, Ryan Tibshirani, and Pradeep Ravikumar. Each with his own style, they have all had great influence in the way I think and do statistical research.

I would like to thank the Statistics and Data Sciences Department (SDS), at The University of Texas at Austin. Its support has made possible every step of my way in the PhD program. In particular, I am very grateful to Michael Daniels and Vicky Keller for making my stay in SDS as smooth as possible.

I also would like to acknowledge the partial support by the CAREER

grant (DMS-1255187) from the U.S. National Science Foundation. Moreover, the motion capture data used in this thesis was obtained from mocap.cs.cmu.edu. The database was created with funding from NSF EIA-0196217.

My sincere gratitude also goes to my colleagues, teachers and friends for making my life in Austin very pleasant.

I would like to thank the dissertation committee members for taking the time to read and help me improve this dissertation.

Finally, to my family. My parents Alejandrina Padilla and Jose Madrid. They have provided everything for me. I thank them for their inconditional love and incredible vision of life. To my siblings, thanks for all their encouragement and love.

Constrained estimation via the fused lasso and some generalizations

Publication No. _____

Oscar Hernan Madrid Padilla, Ph.D.
The University of Texas at Austin, 2017

Supervisor: James G. Scott

This dissertation studies structurally constrained statistical estimators. Two entwined main themes are developed: computationally efficient algorithms, and strong statistical guarantees of estimators across a wide range of frameworks.

In the first chapter we discuss a unified view of optimization problems that enforces constrains, such as smoothness, in statistical inference. This in turn helps to incorporate spatial and/or temporal information about data.

The second chapter studies the fused lasso, a non-parametric regression estimator commonly used for graph denoising. This has been widely used in applications where the graph structure indicates that neighbor nodes have similar signal values. I prove for the fused lasso on arbitrary graphs,

an upper bound on the mean squared error that depends on the total variation of the underlying signal on the graph. Moreover, I provide a surrogate estimator that can be found in linear time and attains the same upper–bound.

In the third chapter I present an approach for penalized tensor decomposition (PTD) that estimates smoothly varying latent factors in multiway data. This generalizes existing work on sparse tensor decomposition and penalized matrix decomposition, in a manner parallel to the generalized lasso for regression and smoothing problems. I present an efficient coordinate-wise optimization algorithm for PTD, and characterize its convergence properties.

The fourth chapter proposes histogram trend filtering, a novel approach for density estimation. This estimator arises from looking at surrogate Poisson model for counts of observations in a partition of the support of the data.

The fifth chapter develops a class of estimators for deconvolution in mixture models based on a simple two-step bin-and-smooth procedure, applied to histogram counts. The method is both statistically and computationally efficient. By exploiting recent advances in convex optimization, we are able to provide a full deconvolution path that shows the estimate for the mixing distribution across a range of plausible degrees of smoothness, at far less cost than a full Bayesian analysis.

Finally, the sixth chapter summarizes my contributions and provides possible directions for future work.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

In recent years there has been an increasing interest in the use of penalized methods for different statistical problems. Penalized methods are characterized by providing models that balance between overfitting and structural constrains on the estimators. This includes, fundamental work in regression analysis, density estimation and dimension reduction. This line of work differs from classical techniques, however, in the use of penalty functions that encourage these estimated solutions to be sparse, structured, or both. As previous work has demonstrated, such regularized estimators usually exhibit a favorable bias-variance tradeoff and can also make the estimated models themselves much more interpretable to practitioners.

In the one-dimensional case, penalized regression problems have been widely studied in the literature. In most previous work in this area, different choices of penalties have proven to be successful for specific applications. One of these celebrated penalties is the fused lasso which constrains solutions to be piecewise constants. This can be a very desirable feature for different statistical problems where there is a natural ordering of the observations. For instance, in protein mass spectroscopy and gene expression

data measured from a microarray, the fused lasso has been used to obtain interpretable results. Another widely used penalty is trend filtering that provides piecewise polynomial solutions. Such smooth estimators have found applications in areas as diverse as image processing and demography.

This dissertation studies different statistical problems exploiting some of the penalized likelihood techniques mentioned above, applying these to specific problems particularly related to non-parametric regression, principal component analysis and density estimation. The goals are twofold. First, we study algorithms, using state of the art techniques from convex optimization, that can provide accurate and fast solutions to the estimation problems mentioned above. Secondly, we aim to study statistical guarantees for our algorithms ensuring that with high probability the solutions provided are close, in some metric sense, to the true parameters.

1.1 A regularized likelihood point of view of estimation

The typical framework of statistical inference consists of estimating an unknown parameter θ from a set of observations y. In this work we study different instances of this problem where the parameter lies in a high-dimensional space, which makes estimation difficult, but there is an special structure that can be exploited to perform estimation.

More precisely, we suppose that data $y \in \mathbb{R}^n$ is given, $n \in \mathbb{N}$, and is

generated as

$$y \sim f(\theta_0), \tag{1.1}$$

where the true parameter satisfies $\theta_0 \in \Theta$, and f is a joint density function indexed by parameters in the space Θ. Moreover, we assume, in all the settings considered here, that the observations y_i, $i \in \{1, \ldots, n\}$, are marginally independent.

In many statistical frameworks, in addition to observing the data, practitioners have some prior knowledge about the structure of the true parameter giving rise to the data. We formalize this by saying that there exists a known function $J : \Theta \to \mathbb{R}$ which satisfies

$$J(\theta_0) \leq c,$$

where c is an unknown positive constant, which satisfies $c << n$. Thus, c is much smaller than the sample size.

In the statistics and machine learning communities, the function J is often referred to as penalty, and it is used to incorporate prior beliefs about the problem in hand. For instance, in certain applications one might want to enforce spatial and/or temporal properties of the data.

The first natural question that arises with settings like the ones described above is: how can the parameter θ_0 be estimated? This is usually important for interpretation, visualization, and also prediction tasks related to the data in hand.

There are two traditional approaches for estimation: Bayesian inference, and regularized likelihood optimization. In this thesis we focus almost exclusively in the latter.

Consider the interpretation of the penalty J as the minus-logarithm of a possibly improper prior. Thus, on the unknown true parameter, we place the prior,

$$P(\theta) \;\propto\; \exp(-\lambda\, J(\theta)), \tag{1.2}$$

for a tuning parameter $\lambda > 0$. Combining both (1.1) with (1.2) leads to a Bayesian model which can, in principle, enable estimation via Markov chain Monte Carlo (MCMC) methods. However, such approach might not generally be tractable nor computationally efficient. Hence, we mainly focus on maximum a posteriori estimators. Thus, estimators arising as

$$\hat{\theta} \;=\; \text{argmin}_{\theta\in\Theta} - \log \mathbb{P}(y \mid \theta) + \lambda\, J(\theta)\,. \tag{1.3}$$

The above framework, in a very general way, encompasses the different estimation problems that concern the interest of this thesis. Our goal is to provide computational efficient algorithms for solving problems of the form (5.5), together with statistical guarantees that ensure validity of our procedures. The specific focus of our study is on:

- **Graph denoising** This is a normal means estimation problem, where the data y comes along with a graph structure G. The true parameter θ_0 lies in $\mathbb{R}^n$, and the underlying assumption is that signal values

corresponding to connected nodes in the graph G tend to have similar values. Loosely speaking, the signal θ_0 is smooth along the graph G, and we make use of this property for estimation purposes.

- **Tensor decomposition** In this setting the data is given in the form of a three dimensional array, $\{y_{i,j,k}\} \in \mathbb{R}^{L \times T \times S}$, with $n = L \times T \times S$. Our goal is to design algorithms to extract low dimensional interpretable features in the presence of noisy measurements.

- **Density estimation** For the classical problems of deconvolution and density estimation, I study adaptive non-parametric estimators, emphasizing the natural assumption that the true model is somehow smooth, but can also have sharp regions.

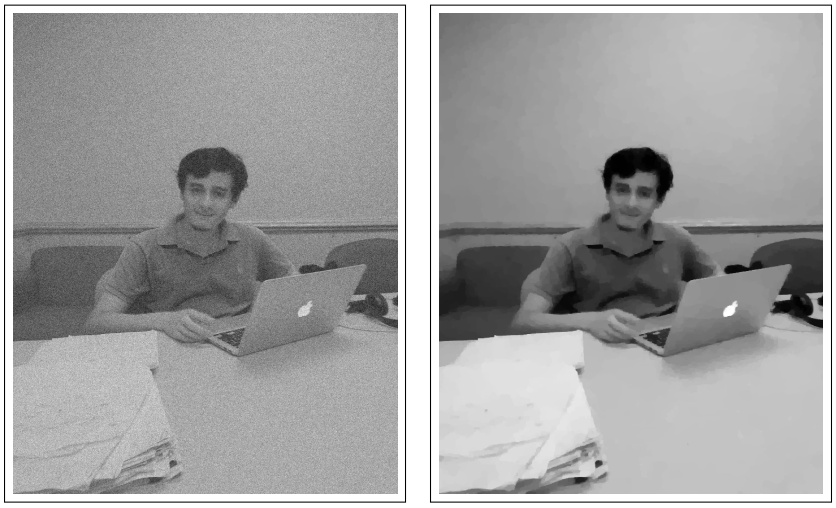
For all of the above research problems, the key to our work is to enforce smoothness assumptions encoded through appropriate penalties J. The penalties we use enjoy adaptivity to different degrees of smoothness.

1.2 Graph denoising

In many applications, one is given noisy measurements on a network, and the goal is to estimate the underlying signal using the network information. The main assumption of graph denoising methods is that the true signal varies smoothly along the graph.

One of the most widely used methods for graph denoising problems, given its attractive computational and statistical properties, is the fused

Figure 1.1: Example of image denoising, the left panel corresponds to a noisy input image, the right panel consists of the fused lasso solution.



lasso estimator [137, 118]. This estimator is obtained as the solution of an optimization problem, as (5.5). In such equation, the likelihood part comes from a Gaussian model, while as we will see in Chapter 2, the penalty J encourages neighboring nodes to have similar signal values.

An example of a graph denoising problem is illustrated in Figure 1.1. There we see a noisy input image and the output of using total variation denoising (fused lasso).

While the fused lasso has attracted great attention for the development of algorithms [28, 74, 8, 136, 90, 87], it is still not known a unifying fast algorithm that can scale well in practice, regardless of the graph. Moreover, it has remained as an open question to understand the convergence rates of the fused lasso on general graphs. In this dissertation we fill this

gap and provide convergence results that hold for any connected graph and any signal. Surprisingly, we will show that such universal guarantees can be attained by a simple procedure that runs in linear time, on the number of edges, for any graph structure.

1.3 Tensor decompositions

Given a data array $Y = \{y_{lts}\}$, practitioners in statistics are often interested in extracting low dimensional factors. This is typically done with the purpose of interpretation and prediction. State of the art methods allow us to do so by using Parafac decompositions. However, an important open question is to provide smooth Parafac decompositions. We address this question by developing convex optimization algorithms that extract smooth low rank decompositions. Such representations can be useful for incorporating spatial and/or temporal information of the data.

The generative model that we study gives rise to a set of observations $y_{l,t,s}$, as

$$y_{l,t,s} = \sum_{j=1}^{J} d_j^* \, u_{lj}^* \, v_{tj}^* \, w_{sj}^* + e_{l,t,s}, \quad , \; l \in \{1, \ldots, L\}, \; t \in \{1, \ldots, T\}, \;\; s \in \{1, \ldots, S\} \tag{1.4}$$

with unknown hidden vectors $u_{:j}^* \in \mathbb{R}^L$, $v_{:j}^* \in \mathbb{R}^T$, $w_{:j}^* \in \mathbb{R}^S$, $j = 1, \ldots, J$ and scalars d_j^*, $j = 1, \ldots, J$. Motivated for interpretation purposes, we make the additional assumption that latent vectors are discrete evaluations of piecewise differentiable functions.

We answer the question of how to estimate these latent factors, by solving a suitable optimization which is a special case of (5.5). The corresponding penalty J is a convex non-differentiable function chosen to encourage sparse and/or smooth factors. However, care needs to be taken as the likelihood is a non-convex function, which together with the non-differentiability of the penalty posses an estimation challenge.

1.4 Density estimation and deconvolution

Density estimators are one of the most commonly used tools for data exploration. They are widely used as first past tool for visualization, and can also be used for predicting future events.

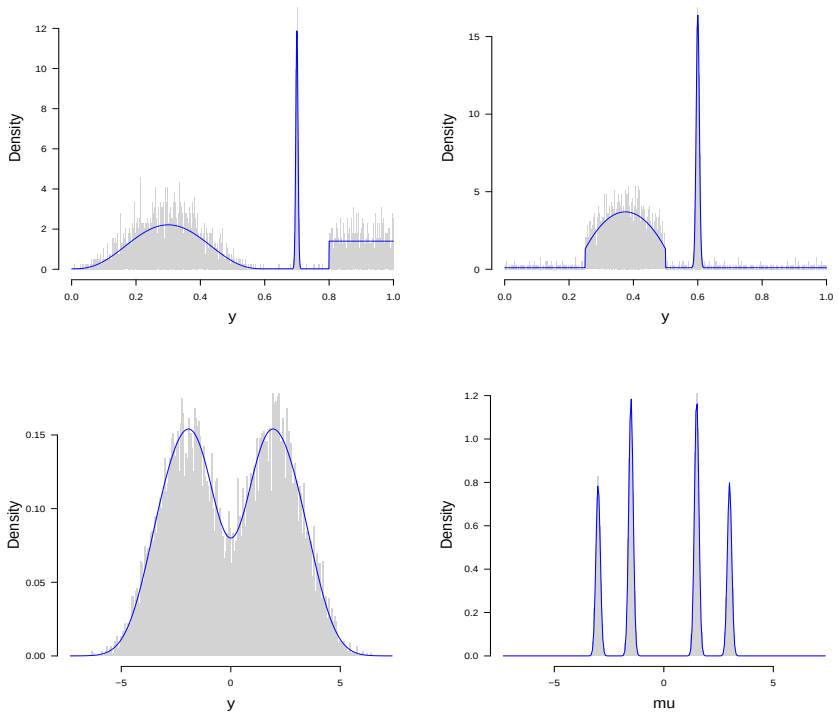
Formally, we study a non-parametric density estimator of a density f_0 that satisfies

$$y_i, \sim^{\text{i.i.d}} f_0, \ i = 1, \dots, n. \tag{1.5}$$

Most practitioners turn to kernel estimators when face with estimating f_0. However, it is well known that kernel estimators cannot properly adapt to different levels of smoothness of the true density. See the first two panels of Figure 1.2 for examples of f_0 that have different levels of smoothness in their domain.

One of the reasons why kernel estimators are widely used in practice is because adaptive estimators are computationally intense. This leads to the natural question: can we provide a computationally efficient adap-

Figure 1.2: The first two panels show two examples of f_0 in model (1.5) plotted on top of the corresponding draws y, which are displayed as a normalized histogram. The panels in the second row show from left to right (for a different f_0), respectively, the density $\phi * f_0$ on top of the draws $\{y_i\}_{i=1}^n$, and the density f_0 on top of the unobserved draws $\{\mu_i\}_{i=1}^n$. Here ϕ is the standard Gaussian density function.



tive density estimator? We propose a possible answer to this question in Chapter 4.

A further difficulty to density estimation in practice occurs when an additional noise level is added to model (1.5). Thus, the data y is actually a

noisy version of unobserved measurements $\{\mu_i\}$ from f_0. Specifically,

$$\begin{array}{lll} y_i \mid \mu_i & \sim^{\text{i.i.d}} & \phi(y_i \mid \mu_i), \\ \mu_i & \sim^{\text{i.i.d}} & f_0. \end{array} \tag{1.6}$$

where ϕ is known distribution. In this modified context, estimating f_0 is known as deconvolution. And although (1.6) seems like a slightly more complicated model than (1.5), it is well known that deconvolution is a much more difficult problem. The author in [56] showed that the convergence rates in deconvolution problems, in ℓ_2 loss function, are in the order of powers of $(\log n)^{-1}$ as opposed to powers of n^{-1}, as it is the case for density estimation.

While the theoretical results, in the way of convergence rates, are very discouraging, deconvolution has been an extensively studied statistical problem [78, 58, 56, 107, 63, 39, 52]. One important question is: how can we construct computationally feasible adaptive deconvolution estimators? This is the focus of Chapter 5, where we propose a novel estimator that also enjoys some desirable statistical properties.

Another important aspect of deconvolution is its connection with the normal means estimation problem, a setting of interest in biological applications (e.g. [48, 123]). This consists in estimating $\{\mu_i\}$ in the case in which ϕ is the standard Gaussian kernel.

The role of deconvolution in the normal means estimation problem is understood through Tweedie's formula which states that

$$E(\mu_i \mid y_i) = y_i + \frac{\mathrm{d}}{\mathrm{d}y} \log m(y_i) = y_i + \frac{m'(y_i)}{m(y_i)}\,, \tag{1.7}$$

where $m(\cdot) := \phi * f_0 = \int \phi(\cdot \mid \mu)\, f_0(\mu)\, d\,\mu$ is the marginal density of the data. Thus, having an estimate of the mixing density (f_0) immediately gives estimates of $\{\mu_i\}$ by a straightforward application of Equation (1.7).

In this work we will provide a deconvolution estimator with strong empirical evidence that can be competitive for the normal means estimation problem. It is beyond the scope of our work to mathematically characterize such performance, but rather to offer a reasonable well behave estimator.

We conclude this section with an example that illustrates the difficulty of deconvolution. This is shown in the bottom two panels of Figure 1.5. The left panel shows the observed data $\{y_i\}$ as a histogram along with the true marginal density. The right panel shows the latent variables $\{\mu_i\}$ as a histogram with the true mixing density. It is clear from these plots that the observed data significantly obscures the actually mixing density, exemplifying how challenging deconvolution can be.

1.5 Total variation penalties

As explained before, the focus of this thesis is to develop efficient algorithms, with statistical guarantees, that overcome some of the challenges and limitations of state of the art methods in the frameworks of graph denoising, low rank tensor decompositions, and deconvolution and density estimation. We do so by proposing estimators that come as the solutions to optimization problems of the form (5.5) using penalties that enforce smooth

solutions. The penalty functions that we use are extensions, or generalizations, of the fused lasso penalty introduced by [118, 102, 137]. This relies on the idea of total variation, which we briefly depict next.

Given a function $f : [0,1] \to \mathbb{R}$, we define its total variation as

$$\text{TV}(f) := \sup \left\{ \sum_{i=1}^{N-1} |f(x_i) - f(x_{i+1})| : x_1 = 0 < x_2 < \ldots < x_N = 1, N \in \mathbb{N} \right\} \tag{1.8}$$

and if f is differentiable, then it can be proven that

$$\text{TV}(f) = \int_0^1 |f'(t)| \, dt.$$

It is clear from its definition that "smoother" functions will tend to have smaller total variation, hence that the total variation offers a natural alternative as regularization in statistical estimation. In such context, it was first introduced by [101] for one-dimensional non-parametric regression. However, one of the possible limitations of using such penalties is that it requires to have an interval, or topological space for the signal's domains, over which partitions are considered and a supremum is taken. This constrains the range of applicability of the TV penalty. This will be more evident in Chapter 2, where we deal with general graph structures. Fortunately, [137] proposed a penalty function, which we refer to as the fussed lasso, that is defined as

$$\text{FL}(\theta) := \sum_{i=1}^{N-1} |\theta_i - \theta_{i+1}|, \quad \forall \theta \in \mathbb{R}^N, \; N \in \mathbb{N}. \tag{1.9}$$

Hence, instead of working with a complicated definition as (1.8), the expression in (1.9) offers a discrete penalty that can easily be generalized to more general contexts, which has led to more computationally efficient algorithms, see [79, 74, 140, 149].

1.6 Outline

The rest of this thesis is organized as follows.

Chapter 2 starts by presenting a brief introduction, followed by a statement of the general graph denoising problem. The chapter then summarizes our contributions before moving to discuss modeling assumptions, notation, and previous work. The latter includes an overview of the literature on the fused lasso focusing in computational and theoretical aspects. All of this constitutes Section 2.1. From there, in Section 2.2, we prove a simple but key lemma relating the ℓ_1 norm (and ℓ_0) norm of differences on a tree and a chain induced by running the depth-first search algorithm (DFS). We then define the DFS fused lasso estimator. In Section 2.3, we derive mean squared error (MSE) rates for the DFS fused lasso, and the fused lasso over the original graph G in question, for signals of bounded variation. We also derive lower bounds for the minimax MSE rate over trees. Section 2.4 proceeds similarly, but for signals with bounded differences. In Section 2.5, we cover numerical experiments, and in Section 2.6, we summarize our work and also describe some potential extensions.

Chapter 3 begins with a discussion of tensor decompositions in Section 3.1, and an overview of previous work in Section 3.2. We then introduce the basic notation on tensors in Section 3.3, which leads to our formulation of rank-1 penalized tensor decomposition (PTD) in Section 3.4. From there, Section 3.5 presents two formulations of our PTD: one with penalties as constrained set, and one with the penalties in the objective function. Section 3.6 provides a convergence analysis for the case of rank-1 decompositions. In Section 3.7 we perform an extensive simulation study highlighting the strengths and weaknesses of our approach. Section 3.8 validates our methods on two real data examples. The first consists of flu hospitalizations in Texas. The second example is based on motion capture data. Finally, Section 3.9 provides a brief discussion of what was presented in the chapter.

In Chapter 4 we study a novel approach for density estimation. Section 4.1 summarizes our contributions. In one dimension, our approach is introduced in Section 4.2. The chapter then discusses, relevant, previous work on non-parametric density estimation. This is the main focus of Section 4.3. It includes aspects such as adaptive and penalized likelihood estimators, log-density estimation, and Lindsey's method. Section 4.4 provides our main theoretical result of the chapter. In Section 4.5, we discuss model selection aspects of our proposed method. Extensions of our one dimensional approach are discussed in Sections 4.6 and 4.7, where we present a Bayesian view and a 2-dimensional estimator respectively. In Section 4.8 we perform one-dimensional validations of our method versus kernel and

adaptive methods in the literature. This section also includes a real data example on NYC taxi data. The conclusion of the chapter is given in Section 4.9.

Chapter 5 studies deconvolution problems. Section 5.1 presents the statistical model of interest, and the relevant literature is discussed in Section 5.2. Our approach is developed with great details in Section 5.3. A sensitivity analysis with real data is performed in Section 5.4. Our consistency result is the main theme of Section 5.6. We then provide simulation experiments in Section 5.6. These consist of examples estimating the mixing density, and also for the normal means estimation problem. We then conclude the chapter with a discussion in Section 5.7.

Finally, our contributions are summarized in Chapter 6, where we also present some open question that are left for future work.

Chapter 2

The DFS Fused Lasso: Linear-Time Denoising over General Graphs

This chapter, based on the working paper [100], studies graph denoising estimators in general graphs. Specifically, the fused lasso, also known as (anisotropic) total variation denoising, which is widely used for piecewise constant signal estimation with respect to a given undirected graph. The fused lasso estimate is highly nontrivial to compute when the underlying graph is large and has an arbitrary structure. But for a special graph structure, namely, the chain graph, the fused lasso—or simply, 1d fused lasso—can be computed in linear time. In this chapter, we establish a surprising connection between the total variation of a generic signal defined over an arbitrary graph, and the total variation of this signal over a chain graph induced by running depth-first search (DFS) over the nodes of the graph. Specifically, we prove that for any signal, its total variation over the induced chain graph is no more than twice its total variation over the original graph. This connection leads to several interesting theoretical and computational conclusions. Denoting by m and n the number of edges and nodes, respectively, of the graph in question, our result implies that for an underlying signal with total variation t over the graph, the fused lasso

achieves a mean squared error rate of $t^{2/3}n^{-2/3}$. Moreover, precisely the same mean squared error rate is achieved by running the 1d fused lasso on the induced chain graph from running DFS. Importantly, the latter estimator is simple and computationally cheap, requiring only $O(m)$ operations for constructing the DFS-induced chain and $O(n)$ operations for computing the 1d fused lasso solution over this chain. Further, for trees that have bounded max degree, the error rate of $t^{2/3}n^{-2/3}$ cannot be improved, in the sense that it is the minimax rate for signals that have total variation t over the tree. Finally, we establish several related results—for example, a similar result for a roughness measure defined by the ℓ_0 norm of differences across edges in place of the the total variation metric.

2.1 Statistical model

We study the graph denoising problem, i.e., estimation of a signal $\theta_0 \in \mathbb{R}^n$ from noisy data

$$y_i = \theta_{0,i} + \epsilon_i, \quad i = 1, \ldots, n, \tag{2.1}$$

when the components of θ_0 are associated with the vertices of an undirected, connected graph $G = (V, E)$.

Throughout, a graph $G = (V, E)$ consists of two sets: a set of nodes V, and a set of edges E. Without loss of generality we assume $V = \{1, \ldots, n\}$. Moreover, E consists of unordered pairs (undirected), called edges, (i, j) with $i, j \in V$. If $(i, j) \in E$, we say that i and j are connected. A path in G is

a sequence $i_1, \ldots, i_K$, for some K, such that $(i_l, i_{l+1}) \in E$ for $l = 1, \ldots, K-1$. The graph G is called connected if there is a path between any two pair of nodes.

Versions of (2.1) arise in diverse areas of science and engineering, such as gene expression analysis, protein mass spectrometry, and image denoising. The problem is also archetypal of numerous internet-scale machine learning tasks that involve propagating labels or information across edges in a network (e.g., a network of users, web pages, or YouTube videos).

Methods for graph denoising have been studied extensively in machine learning and signal processing. In machine learning, graph kernels have been proposed for classification and regression, in both supervised and semi-supervised data settings (e.g., [10, 130, 155, 154]). In signal processing, a considerable focus has been placed on the construction of wavelets over graphs (e.g., [30, 27, 60, 67, 125, 126]). We will focus our study on the *fused lasso* over graphs, also known as (anisotropic) *total variation* denoising over graphs. Proposed by [118] in the signal processing literature, and [139] in the statistics literature, the fused lasso estimate is defined by the solution of a convex optimization problem,

$$\hat{\theta}_G = \underset{\theta \in \mathbb{R}^n}{\operatorname{argmin}} \ \frac{1}{2}\|y - \theta\|_2^2 + \lambda \|\nabla_G \theta\|_1, \tag{2.2}$$

where $y = (y_1, \ldots, y_n) \in \mathbb{R}^n$ the vector of observed data, $\lambda \geq 0$ is a tuning parameter, and $\nabla_G \in \mathbb{R}^{m \times n}$ is the edge incidence matrix of the graph G. Note that the subscript on the incidence matrix ∇_G and the fused lasso

solution $\hat{\theta}_G$ in (2.2) emphasize that these quantities are defined with respect to the graph G. The edge incidence matrix ∇_G can be defined as follows, using some notation and terminology from algebraic graph theory (e.g., [64]). First, we assign an arbitrary orientation to edges in the graph, i.e., for each edge $e \in E$, we arbitrarily select one of the two joined vertices to be the head, denoted e^+, and the other to be the tail, denoted e^-. Then, we define a row $(\nabla_G)_e$ of ∇_G, corresponding to the edge e, by

$$(\nabla_G)_{e,e^+} = 1, \;\; (\nabla_G)_{e,e^-} = -1, \;\; (\nabla_G)_{e,v} = 0 \text{ for all } v \neq e^+, e^-,$$

for each $e \in E$. Hence, for an arbitrary $\theta \in \mathbb{R}^n$, we have

$$\|\nabla_G \theta\|_1 = \sum_{e \in E} |\theta_{e^+} - \theta_{e^-}|.$$

We can see that the particular choice of orientation does not affect the value $\|\nabla_G \theta\|_1$, which we refer to as the *total variation* of θ over the graph G.

2.1.1 Summary of results

We will wait until Section 2.1.3 to give a detailed review of literature, both computational and theoretical, on the fused lasso. Here we simply highlight a key computational aspect of the fused lasso to motivate the main results in this chapter. The fused lasso solution in (2.2), for a graph G of arbitrary structure, is highly nontrivial to compute. For a chain graph, however, the fused lasso solution can be computed in linear time (e.g., using dynamic programming or specialized taut-string methods).

The question we answer is: how can we use this fact to our advantage, when seeking to solve (2.2) over an arbitrary graph? Given a generic graph structure G that has m edges and n nodes, it is obvious that we can define a chain graph by running depth-first search (DFS) over the nodes. Far less obvious is that, for any signal, its total variation over the DFS-induced chain graph never exceeds twice its total variation over the original graph. This fact, which we prove, has the following three notable consequences (the first being computational, and next two statistical).

1. No matter the structure of G, we can denoise any signal defined over this graph in $O(m+n)$ operations: $O(m)$ operations for DFS and $O(n)$ operations for the 1d fused lasso on the induced chain. We call the corresponding estimator—the 1d fused lasso run on the DFS-induced chain—the *DFS fused lasso*.

2. For an underlying signal θ_0 that generates the data, as in (2.1), such that $\theta_0 \in \mathrm{BV}_G(t)$, where $\mathrm{BV}_G(t)$ is the class of signals with total variation at most t, defined in (2.4), the DFS fused lasso estimator has mean squared error (MSE) on the order of $t^{2/3}n^{-2/3}$.

3. For an underlying signal $\theta_0 \in \mathrm{BV}_G(t)$, the fused lasso estimator over the original graph, in (2.2), also has MSE on the order of $t^{2/3}n^{-2/3}$.

The fact that such a fast rate, $t^{2/3}n^{-2/3}$, applies for the fused lasso estimator over *any* connected graph structure is somewhat surprising. It

implies that the chain graph represents the hardest graph structure for denoising signals of bounded variation—at least, hardest for the fused lasso, since as we have shown, error rates on general connected graphs can be no worse than the chain rate of $t^{2/3}n^{-2/3}$.

We also complement these MSE upper bounds with the following minimax lower bound over trees.

4. When G is a tree of bounded max degree, the minimax MSE over the class $\mathrm{BV}_G(t)$ scales at the rate $t^{2/3}n^{-2/3}$. Hence, in this setting, the DFS fused lasso estimator attains the optimal rate, as does the fused lasso estimator over G.

Lastly, we prove the following for signals with a bounded number of nonzero edge differences.

5. For an underlying signal $\theta_0 \in \mathrm{BD}_G(s)$, where $\mathrm{BD}_G(s)$ is the class of signals with at most s nonzero edge differences, defined in (2.5), the DFS fused lasso (under a condition on the spacing of nonzero differences over the DFS-induced chain) has MSE on the order of

$$s(\log s + \log\log n)\log n/n + s^{3/2}/n.$$

When G is a tree, the minimax MSE over the class $\mathrm{BD}_G(s)$ scales as $s\log(n/s)/n$. Thus, in this setting, the DFS fused lasso estimator is only off by a $\log\log n$ factor provided that s is small.

This DFS fused lasso gives us an $O(n)$ time algorithm for nearly minimax rate-optimal denoising over trees. On paper, this only saves a factor of $O(\log n)$ operations, as recent work (to be described in Section 2.1.3) has produced an $O(n \log n)$ time algorithm for the fused lasso over trees, by extending earlier dynamic programming ideas over chains. However, dynamic programming on a tree is (a) much more complex than dynamic programming on a chain (since it relies on sophisticated data structures), and (b) noticeably slower in practice than dynamic programming over a chain, especially for large problem sizes. Hence there is still a meaningful difference—both in terms of simplicity and practical computational efficiency—between the DFS fused lasso estimator and the fused lasso over a generic tree.

For a general graph structure, we cannot claim that the statistical rates attained by the DFS fused lasso estimator are optimal, nor can we claim that they match those of fused lasso over the original graph. As an example, recent work (to be discussed in Section 2.1.3) studying the fused lasso over grid graphs shows that estimation error rates for this problem can be much faster than those attained by the DFS fused lasso (and thus the minimax rates over trees). What should be emphasized, however, is that the DFS fused lasso can still be a practically useful method for any graph, running in linear time (in the number of edges) no matter the graph structure, a scaling that is beneficial for truly large problem sizes.

2.1.2 Assumptions and notation

Our theory will be primarily phrased in terms of the mean squared error (MSE) an estimator $\hat{\theta}$ of the mean parameter θ_0 in (2.1), assuming that $\epsilon = (\epsilon_1, \dots, \epsilon_n)$ has i.i.d. mean zero sub-Gaussian components, i.e.,

$$\mathbb{E}(\epsilon_i) = 0, \quad \text{and} \quad \mathbb{P}(|\epsilon_i| > t) \leq M \exp\big(-t^2/(2\sigma^2)\big), \ \text{all} \ t \geq 0, \tag{2.3}$$

for $i = 1, \dots, n$, and constants $M, \sigma > 0$. The MSE of $\hat{\theta}$ will be denoted, with a slight abuse of notation, by

$$\|\hat{\theta} - \theta_0\|_n^2 = \frac{1}{n}\|\hat{\theta} - \theta_0\|_2^2.$$

(In general, for $x \in \mathbb{R}^n$, we denote its scaled ℓ_2 norm by $\|x\|_n = \|x\|_2/\sqrt{n}$.) Of course, the MSE will depend not only on the estimator $\hat{\theta}$ in question but also on the assumptions that we make about θ_0. We will focus our study on two classes of signals. The first is the *bounded variation class*, defined with respect to the graph G, and a radius parameter $t > 0$, as

$$\mathrm{BV}_G(t) = \{\theta \in \mathbb{R}^n : \|\nabla_G \theta\|_1 \leq t\}. \tag{2.4}$$

The second is the *bounded differences class*, defined again with respect to the graph G, and a now a sparsity parameter $s > 0$, as

$$\mathrm{BD}_G(s) = \{\theta \in \mathbb{R}^n : \|\nabla_G \theta\|_0 \leq s\}. \tag{2.5}$$

We call measure of roughness used in the bounded differences class the *cut metric*, given by replacing the ℓ_1 norm used to define the total variation

metric by the ℓ_0 norm, i.e.,

$$\|\nabla_G \theta\|_0 = \sum_{e \in E} 1\{\theta_{e^+} \neq \theta_{e^-}\},$$

which counts the number of nonzero edge differences that appear in θ. Hence, we may think of the former class in (2.4) as representing a type of weak sparsity across these edge differences, and the latter class in (2.5) as representing a type of strong sparsity in edge differences.

When dealing with the chain graph, on n vertices, we will use the following modifications to our notation. We write $\nabla_{1\mathrm{d}} \in \mathbb{R}^{(n-1)\times n}$ for the edge incidence matrix of the chain, i.e.,

$$\nabla_{1\mathrm{d}} = \begin{bmatrix} -1 & 1 & 0 & \ldots & 0 \\ 0 & -1 & 1 & \ldots & 0 \\ \vdots & & \ddots & \ddots & \\ 0 & 0 & \ldots & -1 & 1 \end{bmatrix}. \tag{2.6}$$

We also write $\hat{\theta}_{1\mathrm{d}}$ for the solution of the fused lasso problem in (2.2) over the chain, also called the *1d fused lasso* solution, i.e., to be explicit,

$$\hat{\theta}_{1\mathrm{d}} = \operatorname*{argmin}_{\theta \in \mathbb{R}^n} \frac{1}{2}\|y - \theta\|_2^2 + \lambda \sum_{i=1}^{n-1} |\theta_{i+1} - \theta_i|. \tag{2.7}$$

We write $\mathrm{BV}_{1\mathrm{d}}(t)$ and $\mathrm{BD}_{1\mathrm{d}}(s)$ for the bounded variation and bounded differences classes with respect to the chain, i.e., to be explicit,

$$\mathrm{BV}_{1\mathrm{d}}(t) = \{\theta \in \mathbb{R}^n : \|\nabla_{1\mathrm{d}}\theta\|_1 \leq t\},$$
$$\mathrm{BD}_{1\mathrm{d}}(s) = \{\theta \in \mathbb{R}^n : \|\nabla_{1\mathrm{d}}\theta\|_0 \leq s\}.$$

Lastly, in addition to the standard notation $a_n = O(b_n)$, for sequences a_n, b_n such that a_n/b_n is upper bounded for n large enough, we use $a_n \asymp b_n$ to

denote that both $a_n = O(b_n)$ and $a_n^{-1} = O(b_n^{-1})$. Also, for random sequences A_n, B_n, we use $A_n = O_{\mathbb{P}}(B_n)$ to denote that A_n/B_n is bounded in probability.

2.1.3 Related work

Since its inception in the signal processing and statistics communities in [118] and [139], respectively, there has been an impressive amount of work on total variation penalization and the fused lasso. We do not attempt to give a complete coverage, but point out some relevant computational and theoretical advances, covering the two categories separately.

Computational. On the computational side, it is first worth pointing out that there are multiple efficient algorithms for solving the fused lasso problem over a chain graph, i.e., the 1d fused lasso problem. The authors in [33] derived an algorithm based on a "taut string" perspective that solves the 1d fused lasso problem in $O(n)$ time (but, the fact that their taut string method solves the 1d fused lasso problem was not explicitly stated in the work). This was later extended by [28, 8] to allow for arbitrary weights in both of the individual penalty and loss terms. The work in [74] proposed an entirely different $O(n)$ time algorithm for the fused lasso based on dynamic programming. The taut string and dynamic programming algorithms are extremely fast in practice (e.g., they can solve a 1d fused lasso problem with n in the tens of millions in just a few seconds on a standard laptop).

More recently, [87] extended the dynamic programming approach of

[74] to solve the fused lasso problem on a tree. This algorithm is theoretically very efficient, with $O(n \log n)$ running time, but the implementation that achieves this running time (we have found) can be practically slow for large problem sizes, compared to dynamic programming on a chain graph. Alternative implementations are possible, and may well improve practical efficiency, but as far as we see it, they will all involve somewhat sophisticated data structures in the "merge" steps in the forward pass of dynamic programming.

The authors in [8] extended (though not in the same direct manner) fast 1d fused lasso optimizers to work over grid graphs, using operator splitting techniques like Douglas-Rachford splitting. Their techniques appear to be quite efficient in practice, and the authors provide thorough comparisons and a thorough literature review of related methods. Over general graphs structures, many algorithms have been proposed, e.g., to highlight a few: [23] described a direct algorithm based on a reduction to parametric max flow programming; [70, 141] gave solution path algorithms (tracing out the solution in (2.2) over all $\lambda \in [0, \infty]$); [24] described what can be seen as a kind of preconditioned ADMM-style algorithm; [88] described an active set approach; [136] leveraged fast 1d fused lasso solvers in an ADMM decomposition over trails of the graph; most recently, [90] derived a new method based on graph cuts. We emphasize that, even with the advent of these numerous clever computational techniques for the fused lasso over general graphs, it is still far slower to solve the fused lasso over an arbitrary

graph than it is to solve the fused lasso over a chain.

Theoretical. On the theoretical side, it seems that the majority of statistical theory on the fused lasso can be placed into two categories: analysis of changepoint recovery, and analysis of MSE. Some examples of works focusing on changepoint recovery are [113, 68, 109, 116]. The statistical theory will concern MSE rates, and hence we give a more detailed review of related literature for this topic.

We begin with results for chain graphs. [102] proved, when $\theta_0 \in \mathrm{BV}_{1\mathrm{d}}(t)$, that the 1d fused lasso estimator estimator $\hat{\theta}_{1\mathrm{d}}$ with $\lambda \asymp t^{-1/3}n^{1/3}$ satisfies

$$\|\hat{\theta}_{1\mathrm{d}} - \theta_0\|_n^2 = O_{\mathbb{P}}(t^{2/3}n^{-2/3}). \tag{2.8}$$

This is indeed the minimax MSE rate for the class $\mathrm{BV}_{1\mathrm{d}}(t)$, as implied by the minimax results in [45]. (For descriptions of the above upper bound and this minimax rate in a language more in line with that of the current paper, see [141].) Recently, [93] improved on earlier results for the bounded differences class in [32], and proved that when $\theta_0 \in \mathrm{BD}_{1\mathrm{d}}(s)$, the 1d fused lasso estimator $\hat{\theta}_{1\mathrm{d}}$ with $\lambda \asymp (nW_n)^{1/4}$ satisfies

$$\|\hat{\theta}_{1\mathrm{d}} - \theta_0\|_n^2 = O_{\mathbb{P}}\bigg(\frac{s}{n}\Big((\log s + \log\log n)\log n + \sqrt{n/W_n}\Big)\bigg), \tag{2.9}$$

where W_n denotes the minimum distance between positions at which nonzero differences occur in θ_0, more precisely,

$$W_n = \min\{|i-j| : (\nabla_{1\mathrm{d}}\theta_0)_i \neq 0, (\nabla_{1\mathrm{d}}\theta_0)_j \neq 0\}.$$

When these nonzero differences or "jumps" in θ_0 are evenly spaced apart, we have $W_n \asymp n/s$, and the above becomes, for $\lambda \asymp \sqrt{n} s^{-1/4}$,

$$\|\hat{\theta}_{1\mathrm{d}} - \theta_0\|_n^2 = O_{\mathbb{P}}\bigg(\frac{s(\log s + \log\log n)\log n}{n} + \frac{s^{3/2}}{n}\bigg). \tag{2.10}$$

This is quite close to the minimax lower bound, whose rate is $s\log(n/s)/n$, that we establish for the class $\mathrm{BD}_{1\mathrm{d}}(s)$, in Theorem 2.4.3. (The minimax lower bound that we prove this theorem actually holds beyond the chain graph, and applies to tree graphs). We can see that the 1d fused lasso rate in (2.10) is only off by a factor of $\log\log n$, provided that s does not grow too fast (specifically, $s = O((\log n \log\log n)^2)$).

Beyond chain graphs, the story is in general much less clear, however, interesting results are known in special cases. For a d-dimensional grid graph, with $d \geq 2$, [71] recently improved on results of [149], showing that for $\theta_0 \in \mathrm{BV}_G(t) \cap \mathrm{BD}_G(s)$, the fused lasso estimator $\hat{\theta}_G$ over G satisfies

$$\|\hat{\theta}_G - \theta_0\|_n^2 = O_{\mathbb{P}}\bigg(\min\{t, s\}\frac{\log^a n}{n}\bigg). \tag{2.11}$$

when $\lambda \asymp \log^{a/2} n$, where $a = 2$ if $d = 2$, and $a = 1$ if $d \geq 3$. A minimax lower bound on the MSE rate for the $\mathrm{BV}_G(t)$ class over a grid G of dimension $d \geq 2$ was established to be $t\sqrt{\log(n/t)}/n$, by [119]. This makes the rate achieved by the fused lasso in (2.11) nearly optimal for bounded variation signals, off by at most a $\log^{3/2} n$ factor when $d = 2$, and a $\log n$ factor when $d \geq 3$.

Other work in [149, 71] also derived MSE rates for the fused lasso over several other graph structures, such as Erdos-Renyi random graphs,

Ramanujan d-regular graphs, star graphs, and complete graphs. As it is perhaps the most relevant to our goals in this chapter, we highlight the MSE bound from [149] that applies to arbitrary connected graphs. Their Theorem 3 implies, for a generic connected graph G, $\theta_0 \in \mathrm{BV}_G(t)$, that the fused lasso estimator $\hat{\theta}_G$ over G with $\lambda \asymp \sqrt{n \log n}$ satisfies

$$\|\hat{\theta}_G - \theta_0\|_n^2 = O_{\mathbb{P}}\bigg(t\sqrt{\frac{\log n}{n}}\bigg). \tag{2.12}$$

(See Appendix A.1 for details.) In Theorem 2.3.2, we show that the universal $tn^{-1/2}$ rate (ignoring log terms) in (2.12) for the fused lasso over an arbitrary connected graph can be improved to $t^{2/3}n^{-2/3}$. In Theorem 2.3.1, we show that the same rate can indeed be achieved by a simple, linear-time algorithm: the DFS fused lasso.

2.2 The DFS fused lasso

In this section, we define the DFS-induced chain graph and the DFS fused lasso.

2.2.1 Tree and chain embeddings

We start by studying some of the fundamental properties associated with total variation on general graphs, and embedded trees and chains. Given a graph $G = (V, E)$, let $T = (V, E_T)$ be an arbitrary spanning tree of G. It is clear that for any signal, its total variation of over T is no larger

than its total variation over G,

$$\|\nabla_T \theta\|_1 = \sum_{e \in E_T} |\theta_{e^+} - \theta_{e^-}| \leq \sum_{e \in E} |\theta_{e^+} - \theta_{e^-}| = \|\nabla_G \theta\|_1, \quad \text{for all } \theta \in \mathbb{R}^n. \tag{2.13}$$

The above inequality, albeit very simple, reveals to us the following important fact: if the underlying mean θ_0 in (2.1) is assumed to be smooth with respect to the graph G, inasmuch as $\|\nabla_G \theta_0\|_1 \leq t$, then it must also be smooth with respect to any spanning tree T of G, since $\|\nabla_T \theta_0\|_1 \leq t$. Roughly speaking, computing the fused lasso solution in (2.2) over a spanning tree T, instead of G, would therefore still be reasonable for the denoising purposes, as the mean θ_0 would still be smooth over T according to the total variation metric.

The same property as in (2.14) also holds if we replace total variation by the cut metric:

$$\|\nabla_T \theta\|_0 = \sum_{e \in E_T} 1\{\theta_{e^+} \neq \theta_{e^-}\} \leq \sum_{e \in E} 1\{\theta_{e^+} \neq \theta_{e^-}\} = \|\nabla_G \theta\|_0, \quad \text{for all } \theta \in \mathbb{R}^n. \tag{2.14}$$

Thus for the mean θ_0, the property $\|\nabla_G \theta_0\|_0 \leq s$ again implies $\|\nabla_T \theta_0\|_0 \leq s$ for any spanning tree T of G, and this would again justify solving the fused lasso over T, in place of G, assuming smoothness of θ_0 with respect to the cut metric in the first place.

Here we go one step further than (2.13), (2.14), and assert that analogous properties actually hold for specially embedded chain graphs. The next lemma gives the key result.

Lemma 2.2.1. *Let $G = (V, E)$ be a connected graph, where recall we write $V = \{1, \ldots, n\}$. Consider depth-first search (DFS) run on G, and denote by $v_1, \ldots, v_n$ the nodes in the order in which they are reached by DFS. Hence, DFS first visits v_1, then v_2, then v_3, etc. This induces a bijection $\tau : \{1, \ldots, n\} \to \{1, \ldots, n\}$, such that*

$$\tau(i) = v_i, \quad \textit{for all } \; i = 1, \ldots, n.$$

Let $P \in \mathbb{R}^{n \times n}$ denote the permutation associated with τ. Then it holds that

$$\|\nabla_{1\mathrm{d}} P\theta\|_1 \leq 2\|\nabla_G \theta\|_1, \quad \textit{for all } \; \theta \in \mathbb{R}^n, \tag{2.15}$$

as well as

$$\|\nabla_{1\mathrm{d}} P\theta\|_0 \leq 2\|\nabla_G \theta\|_0, \quad \textit{for all } \; \theta \in \mathbb{R}^n. \tag{2.16}$$

Proof. The proof is simple. Observe that

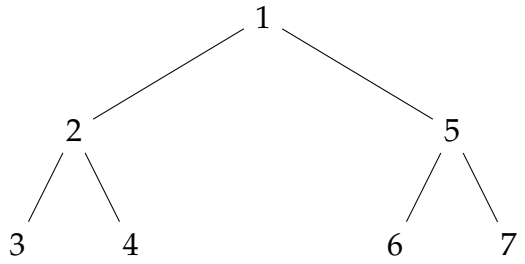
$$\|\nabla_{1\mathrm{d}} P\theta\|_1 = \sum_{i=1,\ldots,n-1} |\theta_{\tau(i+1)} - \theta_{\tau(i)}|, \tag{2.17}$$

and consider an arbitrary summand $|\theta_{\tau(i+1)} - \theta_{\tau(i)}|$. There are now two cases to examine. First, suppose $\tau(i)$ is not a leaf node, and $\tau(i+1)$ has not yet been visited by DFS; then there is an edge $e \in E$ such that $\{e^-, e^+\} = \{\tau(i), \tau(i+1)\}$, and $|\theta_{\tau(i+1)} - \theta_{\tau(i)}| = |\theta_{e^+} - \theta_{e^-}|$. Second, suppose that either $\tau(i)$ is a leaf node, or all of its neighbors have already been visited by DFS; then there is a path $p = \{p_1, \ldots, p_r\}$ in the graph such that $p_1 = \tau(i)$, $p_r = \tau(i+1)$, and each $\{p_j, p_{j+1}\} \in E$, $j = 1, \ldots, r-1$, so that by the triangle inequality

$$|\theta_{\tau(i+1)} - \theta_{\tau(i)}| \leq \sum_{j=1}^{r-1} |\theta_{p_{j-1}} - \theta_{p_j}|.$$

Applying this logic over all terms in the sum in (2.17), and invoking the fundamental property that DFS visits each edge a most twice (e.g., Chapter 22 of [29]), we have established (2.15). The proof for (2.16) follows from precisely the same arguments. □

Example 1. The proof behind Lemma 2.2.1 can also be clearly demonstrated through an example. We consider G to be a binary tree graph with $n = 7$ nodes, shown below, where we have labeled the nodes according to the order in which they are visited by DFS (i.e., so that here P is the identity).



In this case,

$$\begin{aligned}
\|\Delta_{1\mathrm{d}}\theta\|_1 &= \sum_{i=1}^{6} |\theta_{i+1} - \theta_i| \\
&\leq |\theta_2 - \theta_1| + |\theta_3 - \theta_2| + \big(|\theta_3 - \theta_2| + |\theta_4 - \theta_2|\big) + \\
&\big(|\theta_4 - \theta_2| + |\theta_2 - \theta_1| + |\theta_5 - \theta_1|\big) \\
&\quad + |\theta_6 - \theta_5| + \big(|\theta_6 - \theta_5| + |\theta_7 - \theta_5|\big) \\
&\leq 2 \sum_{e \in G} |\theta_{e^+} - \theta_{e^-}| = 2\|\nabla_G \theta\|_1,
\end{aligned}$$

where in the inequality above, we have used triangle inequality for each term in parentheses individually.

2.2.2 The DFS fused lasso

We define the DFS fused lasso estimator, $\hat{\theta}_{\mathrm{DFS}}$, to be the fused lasso estimator over the chain graph induced by running DFS on G. Formally, if τ denotes the bijection associated with the DFS ordering (as described in Lemma 2.2.1), then the DFS-induced chain graph can be expressed as $C = (V, E_C)$ where $V = \{1, \ldots, n\}$ and $E_C = \{\{\tau(1), \tau(2)\}, \ldots, \{\tau(n-1), \tau(n)\}\}$. Denoting by P the permutation matrix associated with τ, the edge incidence matrix of C is simply $\nabla_C = \nabla_{1\mathrm{d}} P$, and the DFS fused lasso estimator is given by

$$\begin{aligned} \hat{\theta}_{\mathrm{DFS}} &= \operatorname*{argmin}_{\theta \in \mathbb{R}^n} \frac{1}{2}\|y - \theta\|_2^2 + \lambda \|\nabla_{1\mathrm{d}} P \theta\|_1 \\ &= P^\top \left(\operatorname*{argmin}_{\theta \in \mathbb{R}^n} \frac{1}{2}\|Py - \theta\|_2^2 + \lambda \sum_{i=1}^{n-1} |\theta_{i+1} - \theta_i| \right). \end{aligned} \tag{2.18}$$

Therefore, we only need to compute the 1d fused lasso estimator on a permuted data vector Py, and apply the inverse permutation operator $P^\top$, in order to compute $\hat{\theta}_{\mathrm{DFS}}$.

Given the permutation matrix P, the computational cost of (2.18) is $O(n)$, since, to recall the discussion in Section 2.1.3, the 1d fused lasso problem (2.7) can be solved in $O(n)$ operations with dynamic programming or taut string algorithms. The permutation P is obtained by running DFS, which requires $O(m)$ operations, and makes the total computation cost of the DFS fused lasso estimator $O(m + n)$.

It should be noted that, when multiple estimates are desired over the

same graph G, we must only run DFS once, and all subsequent estimates on the induced chain require just $O(n)$ operations.

The bounds in (2.15), (2.16) for the DFS chain are like those in (2.13), (2.14) for spanning trees, and carry the same motivation as that discussed above for spanning trees, beneath (2.13), (2.14): if the mean θ_0 is assumed to be smooth with respect to t, insofar as its total variation satisfies $\|\nabla_G \theta_0\|_1 \leq t$, then denoising with respect to C would also be reasonable, in that $\|\nabla_{1\mathrm{d}} P\theta_0\|_1 \leq 2t$; the same can be said for the cut metric. However, it is the rapid $O(m+n)$ computational cost of the DFS fused lasso, and also the simplicity of the dynamic programming and taut string algorithms for the 1d fused lasso problem (2.7), that makes (2.15), (2.16) particularly appealing compared to (2.13), (2.14). To recall the discussion in Section 2.1.3, the fused lasso can in principle be computed efficiently over a tree, in $O(n \log n)$ operations using dynamic programming, but this requires a much more cumbersome implementation and in practice we have found it to be noticeably slower.

2.2.3 Running DFS on a spanning tree

We can think of the induced chain graph, as described in the last section, as being computed in two steps:

(i) run DFS to compute a spanning tree T of G;

(ii) run DFS on the spanning tree T to define the chain C.

Clearly, this is the same as running DFS on G to define the induced chain C, so decomposing this process into two steps as we have done above may seem odd. But this decomposition provides a useful perspective because it leads to the idea that we could compute the spanning tree T in Step (i) in any fashion, and then proceed with DFS on T in Step 2 in order to define the chain C. Indeed, any spanning tree in Step (i) will lead to a chain C that has the properties (2.15), (2.16) as guaranteed by Lemma 2.2.1. This may be of interest if we could compute a spanning tree T that better represents the topology of the original graph G, so that the differences over the eventual chain C better mimicks those over G.

An example of a spanning tree whose topology is designed to reflect that of the original graph is a low-stretch spanning tree. Current interest on low-stretch spanning trees began with the breakthrough results in [54]; most recently, [2] showed that a spanning tree with average stretch $O(\log n \log\log n)$ can be computed in $O(m \log n \log\log n)$ operations. In Section 5.6.1, we investigate low-stretch spanning trees experimentally.

In Section 2.6.4, we discuss a setting in which the fused lasso problem (2.2) has arbitrary penalty weights, which gives rise to a weighted graph G. In this setting, an example of a spanning tree that can be crafted so that its edges represent important differences in the original graph is a maximum spanning tree. Prim's and Kruskal's minimum spanning tree algorithms, each of which take $O(m \log n)$ time [29], can be used to compute a maximum spanning tree after we negate all edge weights.

2.2.4 Averaging multiple DFS estimators

Notice that several DFS-induced chains can be formed from a single seed graph G, by running DFS itself on G with different random starts (or random decisions about which edge to follow at each step in DFS), or by computing different spanning trees T of G (possibly themselves randomized) on which we run DFS, or by some combination, etc. Denoting by $\hat{\theta}^{(1)}_{\mathrm{DFS}}, \hat{\theta}^{(2)}_{\mathrm{DFS}}, \ldots, \hat{\theta}^{(K)}_{\mathrm{DFS}}$ the DFS fused lasso estimators fit to K different induced chains, we might believe that the average estimator, $(1/K)\sum_{k=1}^{K} \hat{\theta}^{(k)}_{\mathrm{DFS}}$, will have good denoising performance, as it incorporates fusion at each node in multiple directions. In Section 5.6.1, we demonstrate that this intuition holds true (at least, across the set of experiments we consider).

2.3 Analysis for signals of bounded variation

Throughout this section, we assume that the underlying mean θ_0 in (2.1) satisfies $\theta_0 \in \mathrm{BV}_G(t)$ for a generic connected graph G. We derive upper bounds on the MSE rates of the DFS fused lasso and the fused lasso over G. We also derive a tight lower bound on the minimax MSE when G is a tree that of bounded degree.

2.3.1 The DFS fused lasso

The analysis for the DFS fused lasso estimator is rather straightforward. By assumption, $\|\nabla_G \theta_0\|_1 \leq t$, and thus $\|\nabla_{1\mathrm{d}} P\theta_0\|_1 \leq 2t$ by (2.15) in Lemma 2.2.1. Hence, we may think of our model (2.1) as giving us i.i.d.

data Py around $P\theta_0 \in \mathrm{BV}_{1\mathrm{d}}(2t)$, and we may apply existing results from [102] on the 1d fused lasso for bounded variation signals, as described in (2.8) in Section 2.1.3. This establishes the following.

Theorem 2.3.1. *Consider a data model* (2.1)*, with i.i.d. sub-Gaussian errors as in* (2.3)*, and $\theta_0 \in \mathrm{BV}_G(t)$, where G is a generic connected graph. Then for any DFS ordering of G yielding a permutation matrix P, the DFS fused lasso estimator $\hat{\theta}_{\mathrm{DFS}}$ in* (2.18)*, with a choice of tuning parameter $\lambda \asymp t^{-1/3}n^{1/3}$, has MSE converging in probability at the rate*

$$\|\hat{\theta}_{\mathrm{DFS}} - \theta_0\|_n^2 = O_{\mathbb{P}}(t^{2/3}n^{-2/3}). \tag{2.19}$$

We note that, if multiple DFS fused lasso estimators $\hat{\theta}^{(1)}_{\mathrm{DFS}}, \hat{\theta}^{(2)}_{\mathrm{DFS}}, \ldots, \hat{\theta}^{(K)}_{\mathrm{DFS}}$ are computed across multiple different DFS-induced chains on G, then the average estimator clearly satisfies the same bound as in (2.19),

$$\left\| \frac{1}{K} \sum_{k=1}^{K} \hat{\theta}^{(k)}_{\mathrm{DFS}} - \theta_0 \right\|_n^2 = O_{\mathbb{P}}(t^{2/3}n^{-2/3}),$$

provided that K is held constant, by the triangle inequality.

2.3.2 The graph fused lasso

Interestingly, the chain embedding result (2.15) in Lemma 2.2.1 is not only helpful for establishing the MSE rate for the DFS fused lasso estimator in Theorem 2.3.1, but it can also be used to improve the best known rate for the original fused lasso estimator over the graph G. In Section 2.1.3, we described a result (2.12) that follows from [149], establishing an MSE

rate of $tn^{-1/2}$ rate (ignoring log terms) for the fused lasso estimator over a connected graph G, when $\|\nabla_G \theta_0\|_1 \leq t$. In fact, as we will now show, this can be improved to a rate of $t^{2/3}n^{-2/3}$, just as in (2.19) for the DFS fused lasso.

[149] present a framework for deriving fast MSE rates for fused lasso estimators based on entropy. They show in their Lemma 9 that a bound in probability on the sub-Gaussian complexity

$$\max_{x \in \mathcal{S}_G(1)} \frac{\epsilon^\top x}{\|x\|_2^{1-w/2}}, \tag{2.20}$$

for some $0 < w < 2$, where $\mathcal{S}_G(1) = \{x \in \text{row}(\nabla_G) : \|\nabla_G x\|_1 \leq 1\}$, leads to a bound in probability on the MSE of the fused lasso estimator $\hat{\theta}_G$ over G. ([149] actually assume Gaussian errors, but their Lemma 9, Theorem 10, Lemma 11, and Corollary 12 still hold for sub-Gaussian errors as in (2.3)). The sub-Gaussian complexity in (2.20) is typically controlled via an entropy bound on the class $\mathcal{S}_G(1)$. Typically, one thinks of controlling entropy by focusing on specific classes of graph structures G. Perhaps surprisingly, Lemma 2.2.1 shows we can uniformly control the sub-Gaussian complexity (2.20) over all connected graphs.

For any DFS-induced chain C constructed from G, note first that $\text{row}(\nabla_G) = \text{span}\{\mathbf{1}\}^\perp = \text{row}(\nabla_C)$, where $\mathbf{1} = (1, \ldots, 1) \in \mathbb{R}^n$ is the vector of all 1s. This, and (2.15) in Lemma 2.2.1, imply that

$$\max_{x \in \mathcal{S}_G(1)} \frac{\epsilon^\top x}{\|x\|_2^{1-w/2}} \leq \max_{x \in \mathcal{S}_C(2)} \frac{\epsilon^\top x}{\|x\|_2^{1-w/2}}.$$

Now, taking $w = 1$,

$$\begin{aligned} \max_{x \in \mathcal{S}_C(2)} \frac{\epsilon^\top x}{\|x\|_2^{1/2}} &= \max_{\substack{x \,:\, \mathbf{1}^\top x = 0, \\ \|\nabla_{1\mathrm{d}} P x\|_1 \leq 2}} \frac{\epsilon^\top x}{\|x\|_2^{1/2}} \\ &= \max_{\substack{x \,:\, \mathbf{1}^\top x = 0, \\ \|\nabla_{1\mathrm{d}} x\|_1 \leq 1}} \frac{2^{-1/2}(P\epsilon)^\top x}{\|x\|_2^{1/2}} \\ &= O_{\mathbb{P}}(n^{1/4}). \end{aligned}$$

The last step (asserting that the penultimate term is $O_{\mathbb{P}}(n^{1/4})$) holds by first noting that $P\epsilon$ is equal in law to ϵ (as we have assumed i.i.d. components of the error vector), and then applying results on the chain graph in Theorem 10, Lemma 11, and Corollary 12 of [149]. Applying Lemma 9 of [149], we have now established the following result.

Theorem 2.3.2. *Consider a data model* (2.1)*, with i.i.d. sub-Gaussian errors as in* (2.3)*, and $\theta_0 \in \mathrm{BV}_G(t)$, where G is a generic connected graph. Then the fused lasso estimator $\hat{\theta}_G$ over G, in* (2.2)*, under a choice of tuning parameter $\lambda \asymp t^{-1/3} n^{1/3}$, has MSE converging in probability at the rate*

$$\|\hat{\theta}_G - \theta_0\|_n^2 = O_{\mathbb{P}}(t^{2/3} n^{-2/3}). \tag{2.21}$$

In a sense, the above theorem suggests that the chain graph is among the hardest graphs for denoising bounded variation signals, since the fused lasso estimator on any connected graph G will achieve an MSE rate in that is at least as good as in the chain rate, if not better. In this vein, it is worth emphasizing that the MSE bound in (2.21) is not tight for certain graph structures; a good example is the 2d grid, where we must compare (2.21) from

the theorem to the known MSE bound in (2.11) from [71], the latter being only log factors from optimal, as shown in [119]. It is natural for the 2d grid graph to consider the scaling $t \asymp \sqrt{n}$ (as argued in [119]), in which case the rates for the fused lasso estimator are $n^{-1/3}$ from Theorem 2.3.2 versus $(\log^2 n)n^{-1/2}$ from [71].

2.3.3 Minimax lower bound over trees

We derive a lower bound for the MSE over the class $\mathrm{BV}_G(t)$ when G is a tree graph. The proof applies Assouad's Lemma [152], over a discrete set of probability measures constructed by a careful partitioning of the vertices of G, that balances both the sizes of each partition element and the number of edges crossing in between partition elements. It is deferred until Appendx A.2.

Theorem 2.3.3. *Consider a data model* (2.1)*, with i.i.d. Gaussian errors* $\epsilon_i \sim N(0, \sigma^2)$*,* $i = 1, \ldots, n$*, and with* $\theta_0 \in \mathrm{BV}_G(t)$*, where* G *is a tree graph, having maximum degree* $d_{\max}$*. Then there exists absolute constants* $N, C > 0$*, such that for* $n/(td_{\max}) > N$*,*

$$\inf_{\hat{\theta}} \sup_{\theta_0 \in \mathrm{BV}_G(t)} \mathbb{E}\|\hat{\theta} - \theta_0\|_n^2 \geq C\left(\frac{t}{\sigma d_{\max}^2 n}\right)^{2/3}. \tag{2.22}$$

The theorem demonstrates that, for trees of bounded degree, such as the chain and balanced d-ary trees, the fused lasso estimator over the tree achieves achieves the minimax rate, as does the DFS fused lasso.

2.4 Analysis for signals with bounded differences

We assume that the underlying mean θ_0 in (2.1) satisfies $\theta_0 \in \mathrm{BD}_G(s)$ for a generic connected graph G. We analyze the MSE of the DFS fused lasso, as well as (a particular formulation of) wavelet denoising over G. We again establish a lower bound on the minimax MSE when G is a tree.

2.4.1 The DFS fused lasso

As it was for the bounded variation case, the analysis for the DFS fused lasso estimator is straightforward. By assumption, $\|\nabla_G \theta_0\|_0 \leq s$, thus $\|\nabla_{1\mathrm{d}} P\theta_0\|_0 \leq 2s$ by (2.16) in Lemma 2.2.1, and we may think of our model (2.1) as having i.i.d. data Py around $P\theta_0 \in \mathrm{BD}_{1\mathrm{d}}(2s)$. Applying an existing result on the 1d fused lasso for bounded differences signals, as described in (2.9), from [93], gives the following result.

Theorem 2.4.1. *Consider a data model* (2.1)*, with i.i.d. sub-Gaussian errors as in* (2.3)*, and $\theta_0 \in \mathrm{BD}_G(s)$, for a connected graph G. Consider an arbitrary DFS ordering of G, that defines a permutation matrix P and the DFS fused lasso estimator $\hat{\theta}_{\mathrm{DFS}}$ in* (2.18)*. Denote by*

$$W_n = \min\{|i-j| : (\nabla_{1\mathrm{d}} P\theta_0)_i \neq 0, (\nabla_{1\mathrm{d}} P\theta_0)_j \neq 0\}$$

the minimum distance between positions, measured along the DFS-induced chain, at which nonzero differences or jumps occur in θ_0. Then, under a choice of tuning parameter $\lambda \asymp (nW_n)^{1/4}$, the DFS fused lasso estimator has MSE converging in

probability at the rate

$$\|\hat{\theta}_{\mathrm{DFS}} - \theta_0\|_n^2 = O_{\mathbb{P}}\bigg(\frac{s}{n}\Big((\log s + \log\log n)\log n + \sqrt{n/W_n}\Big)\bigg). \tag{2.23}$$

Hence, if the s jumps along the DFS chain are evenly spaced apart, i.e., $W_n \asymp n/s$, then for $\lambda \asymp \sqrt{n}s^{-1/4}$,

$$\|\hat{\theta}_{\mathrm{DFSr}} - \theta_0\|_n^2 = O_{\mathbb{P}}\bigg(\frac{s(\log s + \log\log n)\log n}{n} + \frac{s^{3/2}}{n}\bigg). \tag{2.24}$$

An undesirable feature of applying existing 1d fused lasso results for signals with bounded differences, in the above result, is the dependence on W_n in the DFS fused lasso error bound (2.23) (we applied the result (2.9) from [93], but the bounds from [32] also depend on W_n, and as far as we can tell, so should any analysis of the 1d fused lasso for signals with bounded differences). In the 1d setting, assuming that $W_n \asymp n/s$, which says that jumps in θ_0 occur at roughly equally spaced positions, is fairly reasonable; but to assume the same when the jumps are measured with respect to the DFS-induced chain, as we must in order to establish (2.24), is perhaps not. Even if the differences apparent in θ_0 over edges in G are somehow (loosely speaking) spaced far apart, running DFS could well produce an ordering such that jumps in $P\theta_0$ occur at positions very close together. We reiterate that the MSE bounds for the DFS fused lasso for bounded variation signals, in Theorem 2.3.1, do not suffer from any such complications.

2.4.2 Graph wavelet denoising

We compare the performances of the DFS fused lasso and wavelet denoising using spanning tree wavelets, for signals with bounded differences. For spanning tree wavelets, the construction starts with a spanning tree and carefully defines a hierarchical decomposition by recursively finding and splitting around a balancing vertex, which is a vertex whose adjacent subtrees are of size at most half of the original tree; this decomposition is used to construct an unbalanced Haar wavelet basis, as in [128]. In [125], it was shown that for any connected graph G, the constructed wavelet basis $W \in \mathbb{R}^{n \times n}$ satisfies

$$\|W\theta\|_0 \leq \lceil \log d_{\max} \rceil \lceil \log n \rceil \|\nabla_G \theta\|_0, \quad \text{for all } \theta \in \mathbb{R}^n, \tag{2.25}$$

where $d_{\max}$ is the maximum degree of G, and the above holds regardless of choice of spanning tree in the wavelet construction. Now consider the wavelet denoising estimator

$$\hat{\theta}_W = \underset{\theta \in \mathbb{R}^n}{\operatorname{argmin}} \ \frac{1}{2}\|y - \theta\|_2^2 + \lambda \|W\theta\|_1. \tag{2.26}$$

The following is an immediate consequence of (2.25), the fact that the wavelet basis W is orthonormal, and standard results about soft-thresholding (e.g., Lemma 2.8 in [75]).

Theorem 2.4.2. *Consider a data model* (2.1)*, with i.i.d. Gaussian errors* $\epsilon_i \sim N(0, \sigma^2)$*,* $i = 1, \ldots, n$*, and with* $\theta_0 \in \mathrm{BV}_G(t)$*, where* G *is a connected graph,*

having maximum degree $d_{\max}$. Then the spanning tree wavelet estimator $\hat{\theta}_W$ in (2.26)*, with a choice $\lambda \asymp \sqrt{\log n}$, has MSE converging in expectation at the rate*

$$\mathbb{E}\|\hat{\theta}_W - \theta_0\|_n^2 = O\bigg(\frac{s \log d_{\max} \log^2 n}{n}\bigg). \tag{2.27}$$

The result in (2.27) has the advantage over the DFS fused lasso result in (2.23) that it does not depend on a hard-to-interpret quantity like W_n, the minimum spacing between jumps along the DFS-induced chain. But when (say) $d_{\max} \asymp 1$, $s \asymp 1$, and we are willing to assume that $W_n \asymp n$ (meaning the jumps of θ_0 occur at positions evenly spaced apart on the DFS chain), we can see that the spanning tree wavelet rate in (2.27) is just slightly slower than the DFS fused lasso rate in (2.24), by a factor of $\log n / \log\log n$.

While the comparison between the DFS fused lasso and wavelet rates, (2.23) and (2.27), show an advantage to spanning tree wavelet denoising, as it does not require assumptions about the spacings between nonzero differences in θ_0, we have found nonetheless that the DFS fused lasso to performs well in practice compared to spanning tree wavelets, and indeed often outperforms the latter in terms of MSE. Experiments comparing the two methods are presented Section 5.6.1.

2.4.3 Minimax lower bound for trees

We now derive a lower bound for the MSE over the class $\mathrm{BD}_G(s)$ when G is a tree graph. The proof relates the current denoising problem to one of estimating sparse normal means, with a careful construction of the

sparsity set using degree properties of trees. It is deferred until Appendix A.3.

Theorem 2.4.3. *Consider a data model* (2.1)*, with i.i.d. Gaussian errors* $\epsilon_i \sim N(0, \sigma^2)$*,* $i = 1, \ldots, n$*, and with* $\theta_0 \in \mathrm{BD}_G(s)$*, where* G *is a tree. Then there are absolute constants* $N, C > 0$*, such that for* $n/s > N$*,*

$$\inf_{\hat{\theta}} \sup_{\theta_0 \in \mathrm{BD}_G(s)} \mathbb{E}\|\hat{\theta} - \theta_0\|_n^2 \geq C\sigma^2 \frac{s}{n} \log\left(\frac{n}{s}\right). \tag{2.28}$$

The MSE lower bound in (2.28) shows that, when we are willing to assume that $W_n \asymp n/s$ in the DFS-induced chain, the DFS fused lasso estimator is a $\log\log n$ factor away from the optimal rate, provided that s is not too large, namely $s = O((\log n \log\log n)^2)$. The spanning tree wavelet estimator, on the other hand, is a $\log n$ factor from optimal, without any real restrictions on s, i.e., it suffices to have $s = O(n^a)$ for some $a > 0$. It is worth remarking that, for large enough s, the lower bound in (2.28) is perhaps not very interesting, as in such a case, we may as well consider the bounded variation lower bound in (2.22), which will likely be tighter (faster).

2.5 Experiments

In this section we compare experimentally the speed and accuracy of two approaches for denoising signals on graphs: the graph fused lasso, and the fused lasso along the chain graph induced by a DFS ordering. In our experiments, we see that the DFS-based denoiser sacrifices a modest amount in terms of mean squared error, while providing gains (sometimes

considerable) in computational speed. This shows that our main theorem, in addition to providing new insights on MSE rates for the graph fused lasso, also has important practice consequences. For truly massive problems, where the full graph denoising problem is impractical to solve, we may use the linear-time DFS fused lasso denoiser, and obtain a favorable tradeoff of accuracy for speed.

2.5.1 Generic graphs

We begin by considering three examples of large graphs of (more or less) generic structure, derived from road networks in three states: California, Pennsylvania, and Texas. Data on these road networks are freely available at `https://snap.stanford.edu`. In these networks, intersections and endpoints are represented by nodes, and roads connecting these intersections or endpoints are represented by undirected edges; see [92] for more details. For each network, we use the biggest connected component as our graph structure to run comparisons. The graph corresponding to California has $n = 1957027$ nodes and $m = 2760388$ edges, the one for Pennsylvania has $n = 1088092$ nodes and $m = 1541898$ edges, and the graph for Texas has $n = 1351137$ nodes and $m = 1879201$ edges. We compare Laplacian smoothing versus the fused lasso over a DFS-induced chain, on the graphs from the three states. We do not compare with the fused lasso over the original graphs, due to its prohibitive computational cost at such large scales.

We used the following procedure to construct a synthetic signal $\theta_0 \in$

$\mathbb{R}^n$ on each of the road network graphs, of piecewise constant nature:

- an initial seed node v_1 is selected uniformly at random from the nodes $V = \{1, \dots, n\}$ in the graph;
- a component C_1 is formed based on the $\lfloor n/10 \rfloor$ nodes closest to v_1 (where the distance between two nodes in the graph is given by the length of the shortest path between them);
- a second seed node v_2 is selected uniformly at random from $G \setminus C_1$;
- a component C_2 is formed based on the $\lfloor n/10 \rfloor$ nodes closest to v_2 (again in shortest path distance);
- this process is repeated until we have a partition $C_1, \dots, C_{10}$ of the node set V into components of (roughly) equal size, and $\theta_0 \in \mathbb{R}^n$ is defined to take constant values on each of these components.

Note that when constructing each C_i, there might small spurious connected components which we attach to the respective C_i.

In our experiments, we considered 20 values of the total variation for the underlying signal. For each, the signal θ_0 was scaled appropriately to achieve the given total variation value, and data $y \in \mathbb{R}^n$ was generated by adding i.i.d. $N(0, 0.2^2)$ noise to the components of θ_0. For each data instance y, the DFS fused lasso and Laplacian smoothing estimators, the former de-

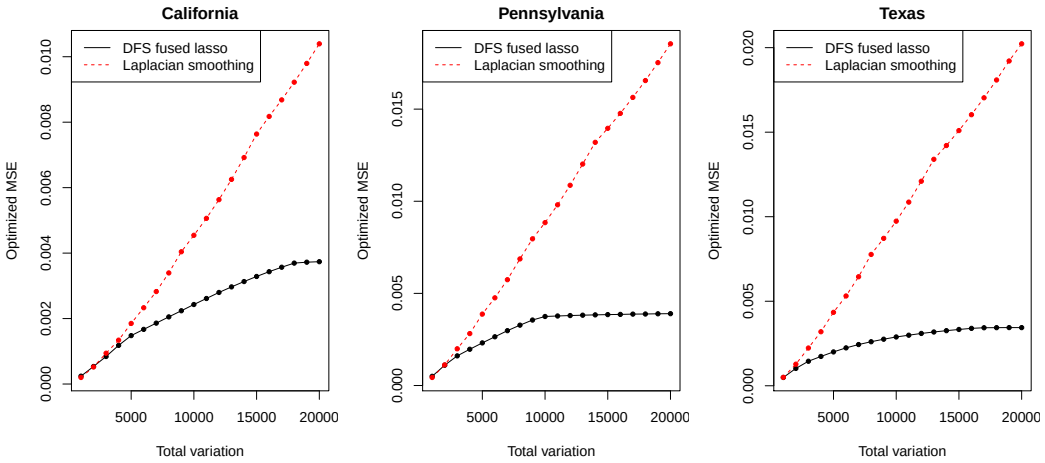
fined by (2.18) and the latter by

$$\hat{\theta}_{\text{Lap}} \;=\; \underset{\theta \in \mathbb{R}^n}{\operatorname{argmin}} \; \frac{1}{2}\|y-\theta\|_2^2 + \lambda \theta^\top L_G \theta, \tag{2.29}$$

where $L_G = \nabla_G^\top \nabla_G$ is the Laplacian matrix of the given graph G, and each estimator is computed over 20 values of its own tuning parameter. Then, the value of the tuning parameter minimizing the average MSE, over 50 draws of data y around θ_0, was selected for each method. Finally, this optimized MSE, averaged over the 50 draws of data y, and further, over 10 repetitions of the procedure for constructing the signal θ_0 explained above, was recorded. Figure 2.1 displays the optimized MSE for the DFS fused lasso and Laplacian smoothing, as the total variation of the underlying signal varies, for the three road network graphs.

As we can see from the figure, for low values of the underlying total variation, i.e., low signal-to-noise ratio (SNR) levels, Laplacian smoothing and the DFS fused lasso, each tuned to optimality, perform about the same. This is because at low enough SNR levels, each will be approximating θ_0 by something like $\bar{y}\mathbf{1}$, with $\bar{y}$ being the sample average of the data vector y. But as the SNR increases, we see that the DFS fused lasso outpeforms Laplacian smoothing by a considerable amount. This might seem surprising, as Laplacian smoothing uses information from the entire graph, whereas the DFS fused lasso reduces the rich structure of the road network graph in each case to that of an embedded chain. However, Laplacian smoothing is a linear smoother (meaning that $\hat{\theta}_{\text{Lap}}$ in (2.29) is a linear function of the data y),

Figure 2.1: *The optimized MSE for the DFS fused lasso and Laplacian smoothing (i.e., MSE achieved by these methods under optimal tuning) is plotted as a function of the total variation of the underlying signal, for each of the three road network graphs. This has been averaged over 50 draws of data y for each construction of the underlying signal θ_0, and 10 repetitions in constructing θ_0 itself. For low values of the underlying total variation, i.e., low SNR levels, the two methods perform about the same, but as the SNR increases, the DFS fused lasso outperforms Laplacian smoothing by a considerable margin.*



and therefore it comes with certain limitations when estimating signals of bounded variation (e.g., see the seminal work of [45], and the more recent graph-based work of [119]). In contrast, the DFS fused lasso is a nonlinear estimator, and while it discards some information in the original graph structure, it retains enough of the strong adaptivity properties of the fused lasso over the original graph to statistically dominate a linear estimator like Laplacian smoothing.

Lastly, in terms of computational time, it took an average of 82.67 seconds, 44.02 seconds, and 54.49 seconds to compute the 20 DFS fused lasso solutions (i.e., over the 20 tuning parameter values) for the road network

graphs from California, Pennsylvania, and Texas, respectively (the averages are taken over the 50 draws of data y around each signal θ_0, and the 10 repetitions in constructing θ_0). By comparison, it took an average of 2748.26 seconds, 1891.97 seconds, and 1487.36 seconds to compute the 20 Laplacian smoothing solutions for the same graphs. The computations and timings were performed on a standard laptop computer (with a 2.80GHz Intel Core i7-2640M processor). For the DFS fused lasso, in each problem instance, we first computed a DFS ordering using the `dfs` function from the R package `igraph`, which is an R wrapper for a C++ implementation of DFS, and initialized the algorithm at a random node for the root. We then computed the appropriate 1d fused lasso solutions using the `trendfilter` function from the R package `glmgen`, which is an R wrapper for a C++ implementation of the fast (linear-time) dynamic programming algorithm in [74]. For Laplacian smoothing, we used the `solve` function from the R package `Matrix`, which is an R wrapper for a C++ implementation of the sparse Cholesky-based solver in [35]. For such large graphs, alternative algorithms, such as (preconditioned) conjugate gradient methods, could certainly be more efficient in computing Laplacian smoothing solutions; our reported timings are only meant to indicate that the DFS fused lasso is efficiently computable at problem sizes that are large enough that even a simple linear method like Laplacian smoothing becomes nontrivial.

Figure 2.2: *Underlying signal, data, and solutions from the 2d fused lasso and different variations on the DFS fused lasso fit over a* 1000×1000 *grid.*

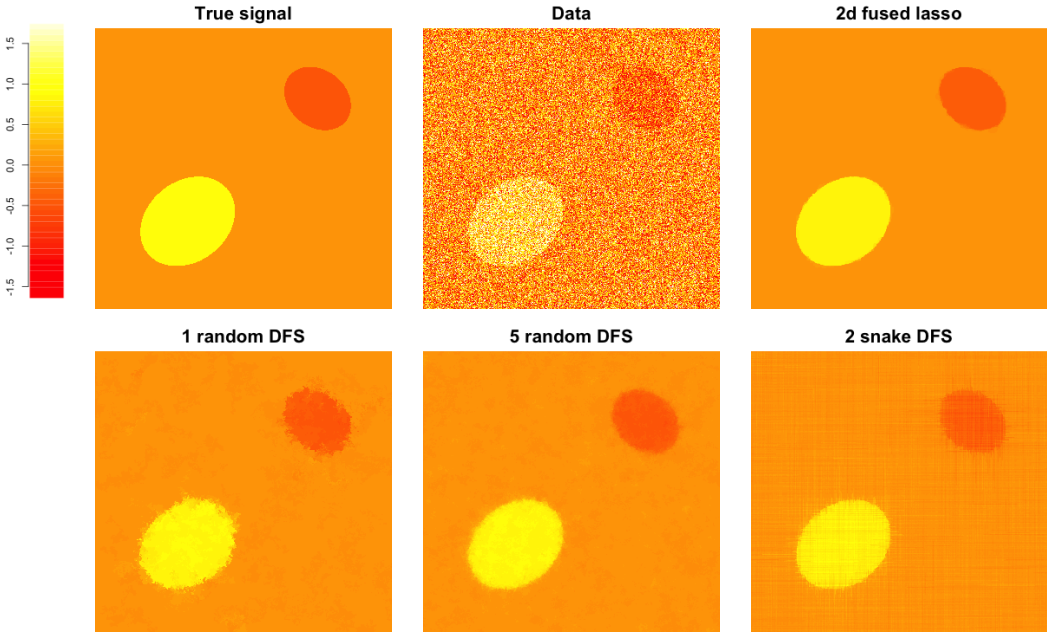
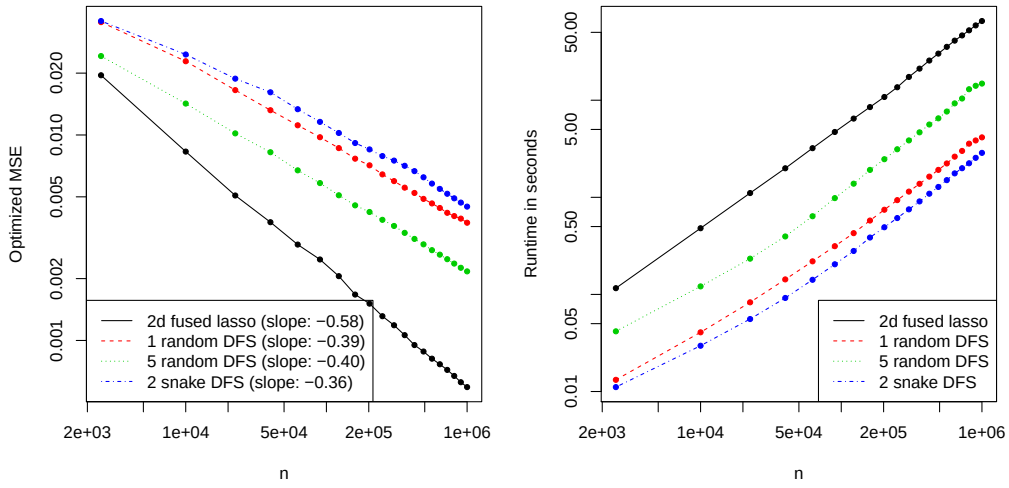


Figure 2.3: *Optimized MSE and runtime for the 2d fused lasso and DFS fused lasso estimators over a 2d grid, as the grid size* n *(total number of nodes) varies.*



2.5.2 2d grid graphs

Next we consider a denoising example on a 2d grid graph of dimension 1000×1000, so that the number of nodes is $n = 1000000$ and the number of edges is $m = 1998000$. We generated a synthetic piecewise constant signal $\theta_0 \in \mathbb{R}^{1000 \times 1000}$ over the 2d grid, shown in the top left corner of Figure 2.2, where a color scale (displayed in the accompanying color legend) is used, with red denoting the smallest possible value and yellow the largest possible value. Data $y \in \mathbb{R}^{1000 \times 1000}$ was generated by adding i.i.d. $N(0, 1)$ noise to the components of θ_0, displayed in the top middle panel of Figure 2.2. We then computed the 2d fused lasso solution (i.e., the fused lasso solution over the full 2d grid graph), as well as three DFS-based variations: the DFS fused lasso solution using a random DFS ordering (given by running DFS beginning at a random node), labeled as "1 random DFS" in the figure; the average of DFS fused lasso solutions over 5 random DFS orderings, labeled "5 random DFS" in the figure; and the average of DFS fused lasso solutions over 2 "snake" DFS orderings (one given by collecting and joining all horizontal edges and the other all vertical edges) labeled "2 snake DFS" in the figure. The tuning parameter for each method displayed in the figure was chosen to minimize the average MSE over 100 draws of the data y from the specified model. Visually, we can see that the full 2d fused lasso solution is the most accurate, however, the 1 random DFS, 5 random DFS, and 2 snake DFS solutions all still clearly capture the structure inherent in the underlying signal. Of the three DFS variations, the 5 random DFS esti-

mator is visually most accurate; the 1 random DFS estimator is comparably "blotchy", and the 2 snake DFS estimator is comparably "stripey".

The left panel of 2.3 shows the optimized MSE for each method, i.e., the minimum of the average MSE over 100 draws of the data y, when we consider 20 choices for the tuning parameter. This optimized MSE is plotted as a function of the sample size, which runs from $n = 2500$ (a 50×50 grid) to $n = 1000000$ (a 1000×1000 grid), and in each case the underlying signal is formed by taking an appropriate (sub)resolution of the image in the top left panel of Figure 2.2. The 2d fused lasso provides the fastest decrease in MSE as n grows, followed by the 5 random DFS estimator, then the 1 random DFS estimator, and the 2 snake DFS estimator. This is not a surprise, since the 2d fused lasso uses the information from the full 2d grid. Indeed, comparing (2.11) and (2.19), we recall that the 2d fused lasso enjoys an MSE rate of $t \log^2 n/n$ when θ_0 has 2d total variation t, whereas the DFS fused lasso has an MSE rate of only $(t/n)^{2/3}$ in this setting. When $t \asymp \sqrt{n}$, which is a natural scaling for the underlying total variation in 2d and also the scaling considered in the experimental setup for the figure, these rates are $(\log^2 n)n^{-1/2}$ for the 2d fused lasso, and $n^{-1/3}$ for the DFS fused lasso. The figure uses a log-log plot, so the MSE curves all appear to have linear trends, and the fitted slopes roughly match these theoretical MSE rates (-0.58 for the 2d fused lasso, and -0.39, -0.40, and -0.36 for the three DFS variations).

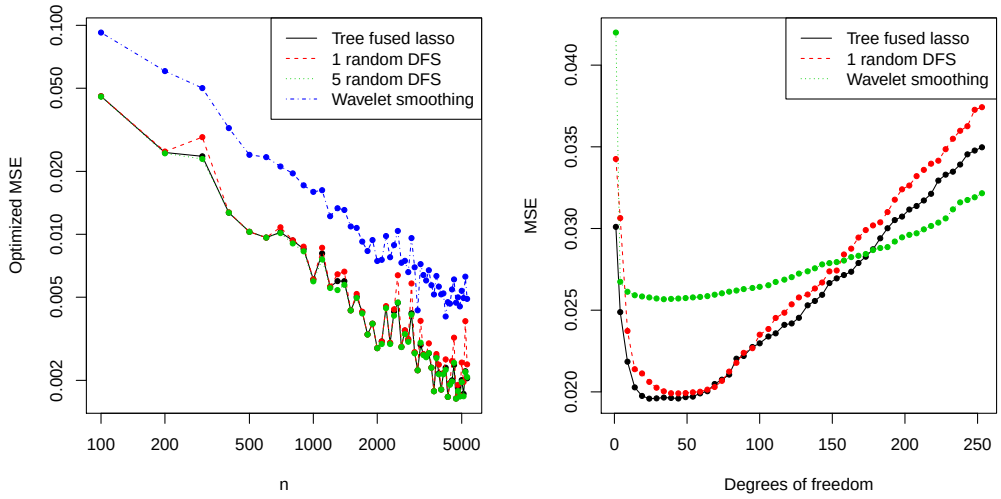
The right panel of Figure 2.3 shows the runtimes for each method

(averaged over 100 draws of the data y), as a function of the sample size n. The runtime for each method counts the total time taken to compute solutions across 20 tuning parameter values. The computations and timings were carried out on a standard desktop computer (with a 3.40GHz Intel Core i7-4770 processor). To compute 2d fused lasso solutions, we used the `TVgen` function in the Matlab package `proxTV`, which is a Matlab wrapper for a C++ implementation of the proximal stacking technique described in [8]. For the DFS fused lasso, we computed initial DFS orderings using the `dfs` function from the Matlab package `MathBGL`, and then, as before, used the C++ implementation available through `glmgen` to compute the appropriate 1d fused lasso solutions. The figure uses a log-log plot, and hence we can see that all DFS-based estimators are quite a bit more efficient than the 2d fused lasso estimator.

2.5.3 Tree graphs

We finish with denoising comparisons on tree graphs, for sample sizes varying from $n = 100$ to $n = 5300$. For each sample size n, a random tree is constructed via a sequential process in which each node is assigned a number of children between 2 and 10 (uniformly at random). Given a tree, an underlying signal $\theta_0 \in \mathbb{R}^n$ is constructed to be piecewise constant with total variation $5\sqrt{n}$ (the piecewise constant construction here is made easy because the oriented incidence matrix of a tree is invertible). Data $y \in \mathbb{R}^n$ was generated by adding i.i.d. $N(0, 1)$ noise to θ_0. We compared the fused

Figure 2.4: *The left panel shows the optimized MSE as a function of the sample size for the fused lasso over a tree graph, as well as the 1 random DFS and 5 random DFS estimators, and wavelet smoothing. The right panel*



lasso estimator over the full tree, 1 random DFS and 5 random DFS estimators (using the terminology from the last subsection), and the wavelet smoothing estimator defined in (2.26). For each estimator, we computed the entire solution path using the path algorithm of [141] implemented in the R package `genlasso`, and selected the step along the path to minimize the average MSE over 50 draws of data y around θ_0, and 10 repetitions in constructing θ_0. (The full solution path can be computed here because each estimator can be cast as a generalized lasso problem, and because the problem sizes considered here are not enormous.)

The left panel of Figure 2.4 plots this optimized MSE as a function of the sample size n. We see that the fused lasso estimator over the full tree and the 5 random DFS estimator perform more or less equivalently

over all sample sizes. The 1 random DFS estimator is slightly worse, and the wavelet smoothing estimator is considerably worse. The right panel shows the the MSE as a function of the effective degrees of freedom of each estimator, for a particular data instance with $n = 5300$. We see that both the tree fused lasso and 1 random DFS estimators achieve their optimum MSEs at solutions of low complexity (degrees of freedom), whereas wavelet smoothing does not come close to achieving this MSE across its entire path of solutions.

2.6 Discussion

Recently, there has been a significant amount on interest on graph-structured denoising. Much of this work has focused on the construction of graph kernels or wavelet bases. We have proposed and studied a simple method, defined by computing the 1d fused lasso over a particular DFS-induced ordering of the nodes of a general graph. This linear-time algorithm comes with strong theoretical guarantees for signals of bounded variation (achieving optimal MSE rates for trees of bounded degree), as well as guarantees for signals with a bounded number of nonzero differences (achieving nearly optimal rates under a condition on the spacings of jumps along the DFS-induced chain). We summarize our theoretical results in Table 2.1.

Practically, we have seen that the DFS fused lasso can often represent a useful tradeoff between computational efficiency and statistical accuracy,

Table 2.1: *A summary of the theoretical results derived in this chapter. All rates are on the mean squared error (MSE) scale ($\mathbb{E}\|\hat{\theta} - \theta_0\|_n^2$ for an estimator $\hat{\theta}$), and for simplicity, are presented under a constant scaling for t, s, the radii in the $\mathrm{BV}_G(t), \mathrm{BD}_G(s)$ classes, respectively. The superscript "$*$" in the $\mathrm{BD}_G(s)$ rate for the DFS fused lasso is used to emphasize that this rate only holds under the assumption that $W_n \asymp n$. Also, we write $d_{\max}$ to denote the max degree of the graph in question.*

	$\mathrm{BV}_G(t)$, $t \asymp 1$	$\mathrm{BD}_G(s)$, $s \asymp 1$
Fused lasso, $\hat{\theta}_G$	$n^{-2/3}$	unknown
Spanning tree wavelets, $\hat{\theta}_W$	unknown	$(\log^2 n \log d_{\max})/n$
DFS fused lasso, $\hat{\theta}_{\mathrm{DFS}}$	$n^{-2/3}$	$(\log n \log\log n)/n^*$
Tree lower bound	$n^{-2/3} d_{\max}^{-4/3}$	$\log n/n$

versus competing methods that offer better statistical denoising power but are more computationally expensive, especially for large problems. A simple trick like averaging multiple DFS fused lasso fits, over multiple random DFS-induced chains, often improves statistical accuracy at little increased computational cost. Several extensions along these lines, and other lines, are possible. To study any of them in detail is beyond the scope of this chapter. We discuss them briefly below, leaving detailed follow-up to future work.

2.6.1 Beyond simple averaging

Given multiple DFS fused lasso estimators, $\hat{\theta}^{(1)}_{\mathrm{DFS}}, \ldots, \hat{\theta}^{(K)}_{\mathrm{DFS}}$, obtained using multiple DFS-induced chains computed on the same graph G, there are several possibilities for intelligently combining these estimators beyond

the simple average, denoted (say) $\bar{\theta}^{(K)}_{\mathrm{DFS}} = (1/K) \sum_{k=1}^{K} \hat{\theta}^{(k)}_{\mathrm{DFS}}$. To better preserve edges in the combined estimator, we could run a simple nonlinear filter—for example, a median filter, over $\hat{\theta}^{(1)}_{\mathrm{DFS}}, \ldots, \hat{\theta}^{(K)}_{\mathrm{DFS}}$ (meaning that the combined estimator is defined by taking medians over local neighborhoods of all of the individual estimators). A more sophisticated approach would be to compute the DFS fused lasso estimators sequentially, using the $(k-1)$st estimator to modify the response in some way in the 1d fused lasso problem that defines the kth DFS fused lasso estimator. We are intentionally vague here with the specifics, because such a modification could be implemented in various ways; for example, it could be useful to borrow ideas from the boosting literature, which would have us treat each DFS fused lasso estimator as a weak learner.

2.6.2 Distributed algorithm

For large graphs, we should be able to both compute a DFS ordering over G, and solve the DFS fused lasso problem in (2.18), in a distributed fashion. There are many algorithms for distributed DFS, offering a variety of communication and time complexities; see, e.g., [144] for a survey. Distributed algorithms for the 1d fused lasso are not as common, though we can appeal to the now well-studied framework for distributed optimization via the alternating direction method of multipliers (ADMM) from [13]. Different formulations for the auxiliary variables present us with different options for communication costs. We have found that, for a formulation that

requires $O(1)$-length messages to be communicated between processors, the algorithm typically converges in a reasonably small number of iterations.

2.6.3 Theory for piecewise constant signals

The bounded differences class $\mathrm{BD}_G(s)$ in (2.5) is defined in terms of the cut metric $\|\nabla_G \theta\|_0$ of a parameter θ, which recall, counts the number of nonzero differences occurring in θ over edges in the graph G. The cut metric measures a notion of strong sparsity (compared to the weaker notion measured by the total variation metric) in a signal θ, over edge differences; but, it may not be measuring sparsity on the "right" scale for certain graphs G. Specifically, the cut metric $\|\nabla_G \theta\|_0$ can actually be quite large for a parameter θ that is piecewise constant over G, with a small number of pieces—these are groups of connected nodes that are assigned the same constant value in θ. Over the 2d grid graph, e.g., one can easily define a parameter θ that has only (say) two constant pieces but on the order of $\sqrt{n}$ nonzero edge differences. Therefore, for such a "simple" configuration of the parameter θ, the cut metric $\|\nabla_G \theta\|_0$ is deceivingly large.

To formally define a metric that measures the number of constant pieces in a parameter θ, with respect to a graph $G = (V, E)$, we introduce a bit of notation. Denote by $Z(\theta) \subseteq E$ the subset of edges over which θ exhibits differences of zero, i.e., $Z(\theta) = \{e \in E : \theta_{e^+} = \theta_{e^-}\}$. Also write $(\nabla_G)_{Z(\theta)}$ for the submatrix of the edge incidence matrix ∇_G with rows in-

dexed by $Z(\theta)$. We define the *piece metric* by

$$\rho_G(\theta) = \mathrm{nullity}\big((\nabla_G)_{Z(\theta)}\big),$$

where $\mathrm{nullity}(\cdot)$ denotes the dimension of the null space of its argument. An equivalent definition is

$$\rho_G(\theta) = \text{the number of connected components in } (V, E \setminus Z(\theta)).$$

We may now define the *piecewise constant class*, with respect to G, and a parameter $s > 0$,

$$\mathrm{PC}_G(s) = \{\theta \in \mathbb{R}^n : \rho_G(\theta) \leq s\}.$$

It is not hard to to see that $\mathrm{BV}_G(s) \subseteq \mathrm{PC}_G(s)$ (assuming only that G is connected), but for certain graph topologies, the latter class $\mathrm{PC}_G(s)$ will be much larger. Indeed, to repeat what we have conveyed above, for the 2d grid one can naturally define a parameter θ such that $\theta \in \mathrm{BD}_G(\sqrt{n})$ and $\theta \in \mathrm{PC}_G(2)$.

We conjecture that the fused lasso estimator over G can achieve a fast MSE rate when the mean θ_0 in (2.1) exhibits a small number of constant pieces, i.e., $\theta_0 \in \mathrm{PC}_G(s)$, provided that these pieces are of roughly equal size. Specifically, assuming $\rho_G(\theta_0) \leq s$, let W_n denote the smallest size of a connected component in the graph $(V, E \setminus Z(\theta_0))$. Then, for a suitable choice of λ, we conjecture that the fused lasso estimator $\hat{\theta}_G$ in (2.2) satisfies

$$\|\hat{\theta}_G - \theta_0\|_n^2 = O_{\mathbb{P}}\bigg(\frac{s}{n}\Big(\mathrm{polylog}\, n + n/W_n\Big)\bigg), \qquad \textbf{(conjecture)}$$

where $\mathrm{polylog}\, n$ is a shorthand for a polynomial of $\log n$. This would substantially improve upon existing strong sparsity denoising results, such as (2.11), (2.23), (2.27), since the latter results are all proven for the class $\mathrm{BD}_G(s)$, which, as we have argued, can be much smaller than $\mathrm{PC}_G(s)$, depending on the structure of G.

2.6.4 Weighted graphs

The key result in Lemma 2.2.1 can be extended to the setting of a weighted graph $G = (V, E, w)$, with $w_e \geq 0$ denoting the edge weight associated an edge $e \in E$. We state the following without proof, since its proof follows in nearly the exact same way as that of Lemma 2.2.1.

Lemma 2.6.1. *Let $G = (V, E, w)$ be a connected weighted graph, where recall we write $V = \{1, \ldots, n\}$, and we assume all edge weights are nonnegative. Consider running DFS on G, and denote by $\tau : \{1, \ldots, n\} \to \{1, \ldots, n\}$ the induced permutation, so that if $v_1, \ldots, v_n$ are the nodes in the order that they are traversed by DFS, then*

$$\tau(i) = v_i, \quad \textit{for all } \; i = 1, \ldots, n.$$

Denote $w_{\min} = \min_{e \in E} w_e$, the minimum edge weight present in the graph, and define

$$\tilde{w}_{ij} = \begin{cases} w_e & \textit{if } e = \{i, j\} \in E, \\ w_{\min} & \textit{otherwise}, \end{cases} \quad \textit{for all } \; i, j = 1, \ldots, n. \tag{2.30}$$

It holds that

$$\sum_{i=1}^{n-1} \tilde{w}_{\tau(i),\tau(i+1)} \big| \theta_{\tau(i+1)} - \theta_{\tau(i)} \big| \leq 2 \sum_{e \in E} w_e |\theta_{e^+} - \theta_{e^-}|, \quad \textit{for all } \; \theta \in \mathbb{R}^n, \tag{2.31}$$

as well as

$$\sum_{i=1}^{n-1} \tilde{w}_{\tau(i),\tau(i+1)} 1\big\{\theta_{\tau(i+1)} \neq \theta_{\tau(i)}\big\} \leq 2 \sum_{e \in E} w_e 1\{\theta_{e^+} \neq \theta_{e^-}\}, \quad \textit{for all } \theta \in \mathbb{R}^n. \tag{2.32}$$

The bounds in (2.31), (2.32) are the analogies of (2.15), (2.16) but for a weighted graph G; indeed we see that we can still embed a DFS chain into G, but this chain itself comes with edge weights, as in (2.30). These new edge weights in the chain do not cause any computational issues; the 1d fused lasso problem with arbitrary penalty weights can still be solved in $O(n)$ time using the taut string algorithm in [8]. Thus, in principle, all of the results in this chapter should carry over in some form to weighted graphs.

2.6.5 Potts and energy minimization

Replacing the total variation metric by the cut metric in the fused lasso problem (2.2) gives us

$$\tilde{\theta}_G = \operatorname*{argmin}_{\theta \in \mathbb{R}^n} \ \frac{1}{2}\|y - \theta\|_2^2 + \lambda \|\nabla_G \theta\|_0, \tag{2.33}$$

often called the *Potts minimization* problem. Because the 1d Potts minimization problem

$$\tilde{\theta}_{1\mathrm{d}} = \operatorname*{argmin}_{\theta \in \mathbb{R}^n} \ \frac{1}{2}\|y - \theta\|_2^2 + \lambda \|\nabla_{1\mathrm{d}} \theta\|_0 \tag{2.34}$$

can be solved efficiently, e.g., in worst-case $O(n^2)$ time with dynamic programming [11, 74], the same strategy that we have proposed in this chapter can be applied to reduce the graph Potts problem (2.33) to a 1d Potts

problem (2.34), via a DFS ordering of the nodes. This may be especially interesting as the original Potts problem (2.33) is non–convex and generally intractable (i.e., intractable to solve to global optimality) for an arbitrary graph structure, so a reduction to a worst-case quadratic-time denoiser is perhaps very valuable.

When the optimization domain in (2.33) is a discrete set, the problem is often called an *energy minimization* problem, as in [15]. It has not escaped our notice that our technique of denoising over DFS-induced chains could be useful for this setting, as well.

Chapter 3

Tensor decomposition with generalized lasso penalties

3.1 Structure and sparsity in multiway arrays

The work presented in this chapter is based on the paper [99]. We study low rank tensor decompositions that are amenable for interpretation purposes. These are particularly useful for incorporating temporal and/or spatial information when analyzing multi–way array data.

Existing methods for penalized matrix decompositions have been shown to outperform classical PCA in discovering patterns in application areas such as genomics and neuroscience. Penalties that encourage structure (such as the fused lasso) provide interpretable results when there is a natural order of the measurements, while penalties that encourage sparsity are useful when there is no such ordering [151]. In the high-dimensional tensor setting however, existing decomposition methods only enforce sparse constraints. We address this gap by proposing a method for penalized tensor decomposition (PTD) that allows arbitrary combinations of sparse or structured penalties along different margins of a data array.

Given a data array $Y = \{y_{lts}\}$, the statistical problem that we study

is to find a low-dimensional factor representation (also known as a Parafac decomposition) such that the factors are constrained to be sparse and/or smooth. For ease of presentation, we restrict attention to the three-way case, but the generalization of our approach to arrays with more than three modes is straightforward.

More explicitly, suppose we are given a set of observations $y_{l,t,s}$, the elements of a three dimensional tensor $\underline{Y} \in \mathbb{R}^{L \times T \times S}$, that have been generated from the complete tensor model

$$y_{l,t,s} = \sum_{j=1}^{J} d_j^* \, u_{lj}^* v_{tj}^* w_{sj}^* + e_{l,t,s}, \quad , \; l \in \{1, \ldots, L\}, \; t \in \{1, \ldots, T\}, \; s \in \{1, \ldots, S\} \tag{3.1}$$

with unknown hidden vectors $u_{:j}^* \in \mathbb{R}^L$, $v_{:j}^* \in \mathbb{R}^T$, $w_{:j}^* \in \mathbb{R}^S$, $j = 1, \ldots, J$ and scalars d_j^*, $j = 1, \ldots, J$. We will later discuss the missing data problem. For simplicity we assume that the variance σ^2 of the error term $e_{l,t,s}$ is constant. Moreover, when $J = 1$ we suppress the index j. Our goal is to estimate these latent factors, which can be challenging since we only have one observation for each combination u_{lj}^*, v_{tj}^*, w_{sj}^*. However, we assume that this task is aided by the presence of special structure in these true vectors. Explicitly, we assume that some of the vectors $\{u_{:,j}^*\}_{j=1}^J$, $\{v_{:,j}^*\}_{j=1}^J$,, $\{w_{:,j}^*\}_{j=1}^J$ are restrictions of smooth functions defined in the interval $[0, 1]$. For instance, it might be the case that $u_{lj}^* = u_j^*(l/L)$ for $l = 1, \ldots, L$, where u_j^* is a piecewise continuous or differentiable function on $[0, 1]$.

A natural situation in which this would arise is when one of the modes of the data array corresponds to a temporal or spatial axis. Our main

contribution is to provide optimization algorithms for finding Parafac decompositions that shrink towards such structure. To do so, we apply a generalized lasso penalty along each mode of the array. We refer to this class of methods as penalized tensor decompositions (PTD).

We face two main challenges in estimating the factors. First, the resulting optimization problem is non-convex. We propose to reach a stationary point using block coordinate descent, as in [3], and we provide convergence rates for a single-block udpate. This leads us to the second challenge: unlike in the sparse unconstrained problem formulated by [3], for our case of a generalized lasso penalty, it is not clear how to make the block-coordinate updates. Our results provide a novel way of doing so that exploits the multi-convex structure of the problem, and that provides efficient algorithms for finding the factors when formulating the problem either in a penalized or constrained form.

3.2 Relation to previous work

Structurally constrained estimation is an active area of research, and we do not attempt a comprehensive review. Our work draws heavily on advances in understanding the one dimensional case, where penalized regression has been widely studied in the literature [59, 80, 138, 139]. For instance, in protein mass spectroscopy and gene expression data measured from a microarray, the fused lasso has been used to obtain interpretable results [139]. The fused lasso is a natural choice here, since it encourages neighboring

measurements to share the same underlying parameter. Similarly, to enforce smoothness in the solution, trend filtering has been proposed by [80] as a way to place one-dimensional function estimation within the convex optimization framework. The trend filtering penalized-regression problem has found applications in areas as diverse as image processing and demography.

In the case of matrix decomposition, the need for penalized methods arises in applications in genetic data, where there are multiple comparative genomic hybridizations and we expect correlation among observations at genetic loci that are close to each other along the chromosome. As shown in [151], by considering different choices of penalties, we can recover different kinds of structures along either the rows or the columns of a data matrix. See the references in [151] for a much more comprehensive bibliography on sparse principal components analysis.

In moving from matrices to multiway arrays, Parafac decompositions offer an attractive framework for recovering latent lower dimensional structure. This is due to their easy interpretability as well as feasibility of computation [4, 69, 76, 86, 89]. More generally, Tucker models have been proposed as general models for multiway data and have been successfully applied in many areas [25]. Other popular methods for tensor decompositions include those described in [12] and [36]. However, these approaches do not provide structural or sparse solutions. This point was made by [3], who proposed a sparse penalized Parafac decomposition method that out-

performs the classical Parafac decomposition when the true solutions are sparse. More recently, [133] also considers sparse tensor recovery and provides statistical guarantees for such a task.

In this chapterf, we study methods for structured, as opposed to sparse, tensor factorizations. Our approach is inspired by the penalized matrix decomposition methods from [151]. We generalize the matrix-decomposition problem to the framework of tensor Parafac decompositions while incorporating solution algorithms for a more broader class of penalties, including trend filtering for factors that are smooth (e.g in space or time).

3.3 Basic definitions

We now introduce notation and definitions used throughout this chapter. This material can be found in [26], to which we refer the reader for more details. Let I_1,I_2..., I_N, denote index N upper bounds. A tensor $\underline{Y} \in \mathbb{R}^{I_1 \times I_2 \times ... \times I_N}$ of order N is an $N-$way array where elements $y_{i_1,i_2...,i_N}$ are indexed by $i_n \in \{1, 2, ..., I_n\}$, for $n = 1, ...,N$. Tensors are denoted by capital letters with a bar, e.g. $\underline{Y} \in \mathbb{R}^{I_1 \times I_2 \times ... \times I_N}$. Matrices are denoted by capital letters, e.g. Y, and for a matrix Y we denote by Y^- its generalized inverse. Vectors are denoted by lower case letters, e.g. y. The outer product of two vectors $a \in \mathbb{R}^I$ and $b \in \mathbb{R}^J$ yields a rank-one matrix $A = a \circ b = ab^T \in \mathbb{R}^{I \times J}$, and the outer product of three vectors $a \in \mathbb{R}^I$, $b \in \mathbb{R}^J$ and $c \in \mathbb{R}^Q$ yields a third-order rank-one tensor$A = a \circ b \circ c \in \mathbb{R}^{I \times J \times Q}$. We use $\| \cdot \|_F$ to indicate the usual Frobenius norm of tensors. The mode-

n multiplication of a tensor $\underline{Y} \in \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_N}$ by a vector $a \in \mathbb{R}^{I_n}$ is denoted by $Z := \underline{Y} \times_n a \in \mathbb{R}^{I_1 \times \ldots \times I_{n-1} \times I_{n+1} \times \ldots \times I_N}$, and element-wise we have $z_{i_1 \ldots i_{n-1} i_{n+1} \ldots i_N} = \sum_{i=1}^{I_n} y_{i_1 i_2 \ldots i_N} a_{i_n}$.

3.4 Penalized tensor decompositions

We first consider the case $J = 1$. Taking a point of view similar to [151], for positive constants c_u, c_v and c_w, we formulate the following problem:

$$\begin{aligned} \underset{u \in \mathbb{R}^L, v \in \mathbb{R}^T, w \in \mathbb{R}^S, g \in \mathbb{R}}{\text{minimize}} \quad & \|Y - g\, u \circ v \circ w\|_F^2 \\ \text{subject to} \quad & \|D^u u\|_1 \leq c_u, \quad \|D^v v\|_1 \leq c_v, \quad \|D^w w\|_1 \leq c_w \\ & u^T u = 1, \quad v^T v = 1, \quad w^T w = 1\,, \end{aligned} \tag{3.2}$$

where D^u, D^v and D^w are matrices which are designed to enforce structural constraints. When the context is clear we will suppress the superscript and simply use the notation D. We note that an alternative, although non-equivalent, formulation is based on an unconstrained version of (3.2) given as

$$\underset{\substack{u^T u = 1,\ v^T v = 1,\\ w^T w = 1}}{\text{minimize}} \quad \|Y - g\, u \circ v \circ w\|_F^2 + \lambda_u \|D^u u\|_1 + \lambda_v \|D^v v\|_1 + \lambda_w \|D^w w\|_1\,, \tag{3.3}$$

with the same unit-norm constraints on the factors. In Section 3, we will discuss the computational differences between these formulations in detail.

We now briefly discuss a broad class of penalties of potential interest to practitioners. We focus on choices that penalize first- and higher-order

differences in each factor, which correspond to the fused lasso and trend filtering, respectively [141]. The fused lasso penalty was suggested in [151] to detect regions of gain for sets of genes in matrix-decomposition problems. For this penalty, the associated D matrix is the $(S-1) \times S$ first-difference matrix, $D_{i,j} = 1$ if $j = i$, $D_{i,j} = -1$ if $j = i+1$ and $D_{i,j} = 0$ otherwise. As discussed in [141], this penalty gives a piecewise-constant solution to linear-regression problems, and it is used in settings where the coordinates in the true model are closely related to their neighbors. Related choices for D are oriented incidence matrices of graphs; see, e.g. [6]. These are constructed as generalizations of the 1-dimensional fused lasso on an underlying graph G.

Still other choices for D correspond to polynomial trend filtering, which imposes a piecewise polynomial structure on the underlying object of interest. These are constructed as follows. First define the polynomial trend filtering of order 1 as $D_{tf,1} \in \mathbb{R}^{(S-2)\times S}$ where $D_{tf,1} = (D^{(1)})^T D^{(1)}$ and $D^{(1)} \in \mathbb{R}^{(S-1)\times S}$ is the first order difference matrix. Then, recursively construct the polynomial trend filtering matrix of order k as $D_{tf,k} = D_{1,d} \cdot D_{tf,k-1}$.

The polynomial trend filtering fits (especially for $k = 3$) are similar to those that one could obtain using regression splines and smoothing splines, However, the knots (changes in kth derivative) in trend filtering are selected adaptively based on the data, jointly with the inter-knot polynomial estimation [141]. A comprehensive study of polynomial trend filtering can be found in [140]. We note that Problem (3.3) was already studied in [3] for the case in which all the matrices D^u, D^v and D^w are set to be the identity.

This is the case of having the L1 penalty on each mode. [3] also proposed a fast algorithm to solve the problem. However, the L1 penalty has the disadvantage of encouraging only sparsity. If the true factors are not sparse, but instead locally flat or smooth, then having sparse constraints on the factors performs poorly. This phenomenon was observed in [151] in the context of matrix decompositions, where the fused lasso penalty was shown to properly recover flat vectors in the factors of the decomposition when the L1 penalty failed to do so. We will extend these ideas to tensor decompositions, applying penalties from the generalized lasso class. We now turn to the question of how to fit these models efficiently.

3.5 Solution algorithms

3.5.1 Constrained problem

Since (3.2) is a non-convex problem, we propose to consider a block coordinate-descent routine. However, in order to have convex block-coordinates-updates, we instead state the following problem:

$$\begin{aligned} \underset{u\in\mathbb{R}^L, v\in\mathbb{R}^T, w\in\mathbb{R}^S}{\text{maximize}} \quad & \underline{Y} \times_1 u \times_2 v \times_3 w \\ \text{subject to} \quad & \|D^u u\|_1 \leq c_u, \quad \|D^v v\|_1 \leq c_v, \quad \|D^w w\|_1 \leq c_w \\ & u^T u \leq 1, \quad v^T v \leq 1, \quad w^T w \leq 1. \end{aligned} \tag{3.4}$$

This differs from (3.2) in two ways. First, the objective has been reformulated in a more convenient way, but it is easy to show that this results in an equivalent problem [86]. Secondly, the unit norm constraints have been relaxed to the convex constraints that each factor falls into the unit ball.

Additionally, following [151], a simple modification can naturally handle missing data. Denoting by M the set missing observations, we solve the missing data problem by replacing the objective function in (3.4) with the function

$$F(u,v,w) = \sum_{(l,t,s)\in\{1,\ldots,L\}\times\{1,\ldots,T\}\times\{1,\ldots,S\}-M} Y_{l,t,s}\, u_l\, v_t\, w_s \tag{3.5}$$

Note that (3.4) has a multilinear objective function in u, v, and w. Since the penalties induced by D^u, D^v and D^w are convex, we can use coordinate-wise optimization in order to solve this problem. For example, when v and w are fixed, the update for u is found by solving the following problem:

$$\underset{u}{\text{maximize}} \;\; (\underline{Y} \times_2 v \times_3 w)^T u \qquad \text{subject to} \quad \|u\|_2^2 \leq 1\,,\; \|D^u u\|_1 \leq c_u. \tag{3.6}$$

It would seem that a solution to (3.6) would not in general have unit norm. But it is possible to ensure that this will be the case—that is, to ensure the solution falls on the boundary of the ℓ^2 constraint set—as long as c_u is chosen properly based on the KKT conditions. A similar phenomenon was observed for the matrix case in [151]. One of our results is that the solution to (3.6) will very often turn out to have unit norm, despite our convex relaxation. A rigorous statement of this result will be given later.

Our strategy to solve (3.4) is to sweep through the vectors iteratively by proceeding with block coordinates updates. Thus starting from initials u^0, v^0 and w^0, we proceed by solving, at iteration m, the problems shown

Algorithm 1 Constrained problem block coordinate descent

$$
\begin{aligned}
u^m &= \arg\min_u \Big\{ \left(-\underline{Y} \times_2 v^{m-1} \times_3 w^{m-1}\right)^T u \\
&\qquad \text{subject to} \quad \|u\|_2^2 \leq 1\,,\ \|D^u u\|_1 \leq c_u. \Big\} \\
v^m &= \arg\min_v \Big\{ \left(-\underline{Y} \times_1 u^m \times_3 w^{m-1}\right)^T v \\
&\qquad \text{subject to} \quad \|v\|_2^2 \leq 1\,,\ \|D^v v\|_1 \leq c_v. \Big\} \\
w^m &= \arg\min_w \Big\{ \left(-\underline{Y} \times_1 u^m \times_2 v^m\right)^T w \\
&\qquad \text{subject to} \quad \|w\|_2^2 \leq 1\,,\ \|D^w w\|_1 \leq c_w. \Big\}
\end{aligned}
$$

in Algorithm 1. It should be pointed out here that the best we can hope with Algorithm 1 is to obtain a local minimum to (3.4). It will be shown latter with our experiments that this local minimum provides interpretable and accurate estimators. Note that while the algorithm is structurally quite simple, the individual block-coordinate updates are non-trivial to solve efficiently. The remainder of this section discusses how this can be done.

Given the symmetry of the problem, without loss of generality, we focus on the update for u. We notice that the constraint set involves a non-differentiable function, implying that it is not possible to use a gradient-based method. Before describing our approach, we first discuss two natural possibilities and explain why they were ultimately rejected.

First, a simple approach is to include a slack variable $z = D^u u$ and use the ADMM algorithm. However, the resulting update for u would require solving a constrained problem using, for example, an interior-point

method. This rapidly becomes infeasible, since it requires solving a large dense linear system.

A second natural approach is to use the novel ADMM algorithm from [156] to solve each of the block-coordinate updates. For instance, the update for u would involve solving the problem

$$\begin{aligned} u^m \;=\; & \underset{u}{\arg\min} \quad (-\underline{Y} \times_2 v^{m-1} \times_3 w^{m-1})^T u \\ & \text{subject to} \quad \|u\|_2^2 \leq 1\,,\ \|z\|_1 \leq c_u, \\ & \qquad\qquad z = D^u u, \quad (E_u - (D^u)^T D^u)^{1/2} u = \tilde{z}\,, \end{aligned} \tag{3.7}$$

where E_u is a matrix such that $E_u \succeq (D^u)^T D^u$. Then proceeding as in [156], we observe that (3.7) the update for u is a simple projection on the unit ℓ_2 ball, while the update for z requires projecting in a ℓ_1 ball with the algorithm from [47]. (The actual updates for our problem are given in the supplementary material.) However, while this algorithm indeed solves the constrained-problem updates, we find in that practice the ADMM routine requires a long time to converge. In particular, it presents problems enforcing the constraint that $\|D^u u^m\|_1 \leq c_u$, so that the solution returned after reasonable runtimes is actually quite far from the feasible region.

This motivates us to consider a different approach to solve the block-coordinate updates in (1). We appeal to the following theorem, which suggests a simple method and also implies that, typically, the solution lies on the boundary of the unit ball. That is, it satisfies the non-convex constraint of problem (3.2), despite our relaxation.

Theorem 3.5.1. *Assume that* $c_u > 0$ *and* $\underline{Y} \times_2 v \times_3 w \notin Range\left((D^u)^T\right)$. *Then the solution to (3.6) is given by*

$$u^* = \frac{\left(\underline{Y} \times_2 v \times_3 w - (D^u)^T \hat{\gamma}_{\lambda^*}\right)}{\|\underline{Y} \times_2 v \times_3 w - (D^u)^T \hat{\gamma}_{\lambda^*}\|_2} \tag{3.8}$$

where

$$\hat{\gamma}_\lambda = \underset{\|\gamma\|_\infty \leq \lambda}{\arg\min} \quad \frac{1}{2}\|\underline{Y} \times_2 v \times_3 w - (D^u)^T \gamma\|_2^2 \tag{3.9}$$

$$\lambda^* = \underset{0 \leq \lambda}{\arg\min} \left[\|\underline{Y} \times_2 v \times_3 w - (D^u)^T \hat{\gamma}_\lambda\|_2 + \lambda c_u\right]. \tag{3.10}$$

Proof. See Appendix B.3.1. □

As a direct consequence of the proof of Theorem 3.5.1, we can solve (3.6) by first solving (3.9) with the solution-path algorithm from [141], then finding λ^* and finally u^*. The explicit algorithm is given in the supplementary material.

Unfortunately, there is no characterization available of the computational time to compute the solution path. It is only known the cost at each iteration is $O(L)$ in its worst case, but it is unknown how many kinks K that a particular problem will have. Moreover, we notice that after the solution path is computed, the next two steps require $O(KL)$ cost. Therefore, the total cost for updating u is $O(KL)$.

3.5.2 Unconstrained version

The framework we have introduced for rank-1 approximations has some nice features. In particular, the choice of tuning parameters is more

intuitive, since this directly imposes a constraint on the smoothness of the solutions. However, the optimization routine derived from Theorem 3.5.1 is computationally intensive. In particular, for large dimensions of the penalty matrices, computing the entire solution path can still be somewhat slow. To avoid this, we revisit (3.3) and consider a problem equivalent to its convex relaxation:

$$\begin{aligned} &\underset{u\in\mathbb{R}^L, v\in\mathbb{R}^T, w\in\mathbb{R}^S}{\text{minimize}} \quad -\underline{Y}\times_1 u\times_2 v\times_3 w + \lambda_u \|D^u u\|_1 + \lambda_v \|D^v v\|_1 + \lambda_w \|D^w w\|_1 \\ &\text{subject to} \qquad u^T u \le 1, \quad v^T v \le 1, \quad w^T w \le 1\,. \end{aligned} \tag{3.11}$$

As in the constrained case, we solve (3.11) via block-coordinate updates. Now the update for u is obtained by solving

$$\underset{u}{\text{minimizing}} \quad -\left(\underline{Y}\times_2 v\times_3 w\right)^T u + \lambda_u \|D^u u\|_1 \qquad \text{subject to} \quad \|u\|_2^2 \le 1\,. \tag{3.12}$$

The solution to (3.12) can be characterized in the same manner as for the constrained case. In fact, the proof of Theorem 3.5.1 automatically implies the following corollary:

Corollary 3.5.2. *With the notation and assumptions from Theorem (3.5.1), the solution to*

$$\underset{u\in\mathbb{R}^S}{\textit{minimize}} -\left(\underline{Y}\times_2 v\times_3 w\right)^T u + \lambda \|D^u u\|_1 \quad \textit{subject to} \quad \|u\|_2^2 \le 1 \tag{3.13}$$

has the following form, where $\hat{\gamma}_\lambda$ is defined in (3.9):

$$u^* = \frac{\left(\underline{Y}\times_2 v\times_3 w - (D^u)^T\hat{\gamma}_\lambda\right)}{\|\underline{Y}\times_2 v\times_3 w - (D^u)^T\hat{\gamma}_\lambda\|_2}\,. \tag{3.14}$$

An interesting consequence of the closed-form formula (3.14), and the proof of Theorem 3.5.1, is that we can solve (3.12) by first solving a generalized lasso problem and then projecting the solution into the unit ball. Explicitly, we first find

$$\hat{u} = \underset{u \in \mathbb{R}^L}{\arg\min} \left\{ \|u - \underline{Y} \times_2 v \times_3 w\|_2^2 + \lambda \|D^u u\|_1 \right\}, \tag{3.15}$$

and $\hat{u}/\|\hat{u}\|_2$ becomes the solution to (3.12). Therefore, for trend-filtering problems, we can solve the regression problem step with the fast ADMM algorithm from [110]. Moreover, for the case of a fused lasso penalty, the update for u can be done in linear time [74]. Because these two algorithms are so efficient, the penalized formulation from (3.11) can be solved much more cheaply than the constrained formulation from (3.4).

3.5.3 A toy example

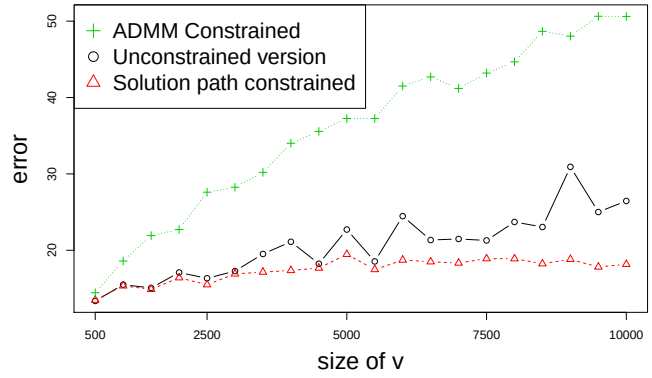
We illustrate the advantage of problem (3.11) over the formulation from (3.4) using a toy example. We consider $u^* \in \mathbb{R}^{10}$ and $w^* \in \mathbb{R}^{400}$ as the size of v^* varies. Here, u^* and w^* are as in Structure 2 in Figure 3.2, while v^* is the function $\cos(9\,\pi\,t)$ evaluated at evenly spaced locations in $[0, 1]$. Taking initial values from the power method, we compare the solutions from one iteration of the unconstrained formulation when choosing the penalty parameters adaptively, versus an "oracle" version of the constrained problem with $(c_u, c_v, c_w) = (\|D^u u^*\|_1, \|D^v v^*\|_1, \|D^w w^*\|_1)$. This choice of hyperparameters for the constrained problem is obviously optimal, but requires knowledge of the true factors, and is therefore unrealistic in practice.

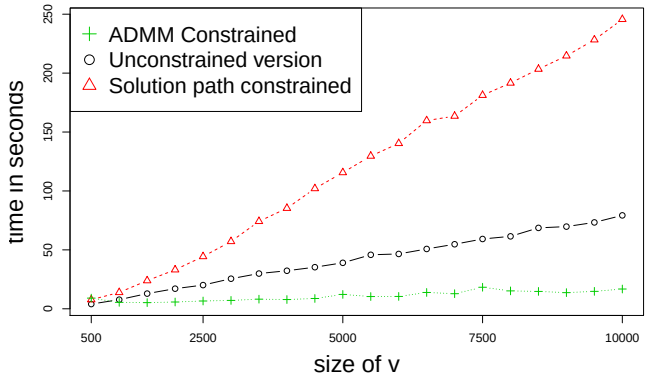
Figure 3.1 demonstrates the favorable trade-off offered by the unconstrained formulation with adaptively chosen tuning parameters. We observe that while the constrained formulation algorithm based on the solution-path computation is the most accurate, the unconstrained formulation is competitive in terms of reconstruction error, and much more efficient. The ADMM algorithm based on [156] is substantially less accurate than the other two methods.

Moreover, in practice it would be necessary to solve the constrained problem with more than one value of the tuning parameters, since we do not know $\|D^u u^*\|_1$. Hence the penalized version is strongly preferred: we can do adaptive parameter choice more cheaply than solving the constrained version for a single hyperparameter setting, without a major loss of performance even under an optimal hyperparameter choice.

With regards to the choice of regularization, we can consider two alternatives based on cross validation. The first of these follows [151]. This procedure involves randomly deleting a percentage of the input data and solves the problem on the resulting tensor. The estimated tensor produces predicted values on the deleted entries, allowing one to compute mean square error of prediction for these notionally missing values. The parameters λ_u, λ_v and λ_w are then chosen to minimize the prediction error. This is particularly attractive when multiple processors are available, given that independent problems with different tuning parameters can be solved in parallel.

Figure 3.1: Panel (a): Frobenius error comparison of the of three different methods for finding a rank-1 decomposition. These are: Algorithm 1 with the ADMM method from [156], block coordinate descent for solving the unconstrained problem (3.11), and Algorithm 1 using the solution path method as described in Section 3.1. Panel (b): For each of the methods, time in seconds for solving one problem with a particular choice of tuning parameters. Our unconstrained formulation with adaptive chosen penalties achieves nearly the reconstruction error of the unconstrained formulation with optimal hyperparameter choice, but at far less computational cost.





The other alternative for cross validation applies to (3.11) and it is based on adaptively choosing the tuning parameters. Thus, before estimat-

ing each vector (say u), we obtain a generalized lasso regression problem and hence we can choose λ_u by cross validation. We randomly separate the coordinates of the response vector into training and test set, solving the problem in the training set and computing the mean squared error of the predicted solution on the test set. This exploits the fact that u is a smooth function, and therefore given a solution based on the training set, we can provide estimates at the locations in the test set by interpolation.

3.5.4 Multiple factors

In the case of multiple factors, the main difference of the tensor case versus the matrix case is that we must find all the factors jointly [86], as opposed to estimating factor $k+1$ using the residual from the fitted k-factor model. Fortunately, it is straightforward to use any of the algorithms in the previous section to handle multiple factors. Hence, to estimate the factors in (3.1), we state the problem

$$\begin{aligned} \underset{u_j, v_j, w_j}{\text{minimize}} \quad & \|\underline{Y} - \sum_{j=1}^{J} d_j \, u_j \circ v_j \circ w_j\|_{\mathrm{F}}^2 \quad + \\ & \sum_{j=1}^{J} \left[\lambda_{u,j} \|D_j^u \, u_j\|_1 + \lambda_{v,j} \|D_j^v \, v_j\|_1 + \lambda_{w,j} \|D_j^w \, w_j\|_1 \right] \\ \text{subject to} \quad & \|u_j\|_2^2 \leq 1 \quad \|v_j\|_2^2 \leq 1 \quad \|w_j\|_2^2 \leq 1 \quad j = 1, ..., J, \end{aligned} \tag{3.16}$$

where the matrices D_j^u,D_j^v and D_j^w are chosen to capture different structural features desired for the solutions. Here, $\lambda_{u,j}$, $\lambda_{v,j}$ and $\lambda_{w,j}$ are tuning parameters. Now we solve (3.16) by starting with initial guesses $\{u^j\}$, $\{v^j\}$, $\{w^j\}$, $\{d^j\}$ and applying the iterative updates listed in Algorithm 2 exploiting re-

Algorithm 2 Multiple factors

Loop for $j_0 = 1 : J$,

$$
\begin{aligned}
u^{j_0} &\leftarrow \underset{\|u\|_2^2 \leq 1}{\arg\min} \Big\{ \Big\| u - \underline{Y} \times_2 v^{j_0} \times_3 w^{j_0} + \textstyle\sum_{j \neq j_0} d^j \, (v^{j_0})^T \, v^j \, (w^{j_0})^T \, w_j \, u^j \Big\|_2^2 + \\
&\qquad \lambda_{u,j_0} \, \|D^u_{j_0} \, u\|_1, \Big\} \\
v^{j_0} &\leftarrow \underset{\|v\|_2^2 \leq 1}{\arg\min} \Big\{ \Big\| v - \underline{Y} \times_1 u^{j_0} \times_3 w^{j_0} + \textstyle\sum_{j \neq j_0} d^j \, (u^{j_0})^T \, u^j \, (w^{j_0})^T \, w_j \, v^j \Big\|_2^2 + \\
&\qquad \lambda_{v,j_0} \, \|D^v_{j_0} \, v\|_1, \Big\} \\
w^{j_0} &\leftarrow \underset{\|w\|_2^2 \leq 1}{\arg\min} \Big\{ \Big\| w - \underline{Y} \times_2 u^{j_0} \times_3 v^{j_0} + \textstyle\sum_{j \neq j_0} d^j \, (u^{j_0})^T \, u^j \, (v^{j_0})^T \, v_j \, w^j \Big\|_2^2 + \\
&\qquad \lambda_{w,j_0} \, \|D^w_{j_0} \, w\|_1. \Big\} \\
d^{j_0} &\leftarrow \underline{Y} \times_1 u^{j_0} \times_2 v^{j_0} \times_3 w^{j_0} - \textstyle\sum_{j \neq j_0} d^j \, (u^{j_0})^T \, u^j \, (v^{j_0})^T \, v^j \, (w^{j_0})^T \, w^j.
\end{aligned}
$$

End loop

sults from Section 3.2.

In practice the number of latent factors can be chosen with an ad-hoc rule by looking at the proportion of the variance explained (as with a scree plot in ordinary PCA). One can look at the solutions provided by different values of J. The choice of J then corresponds to the number factors such that the increase in variance explained that is obtained by solving the problem with more factors is negligible. We illustrate this in our real data example.

Finally, in situations where the number of factors is large, the number of possible combinations of tuning parameters becomes challenging. One possibility to address this is to choose the parameters adaptively as dis-

cussed in Section 3.2. Hence, every time a factor is to be updated we select the parameter from a small grid of values. This ensures that, for instance, when dealing with fused lasso penalties each block coordinated update can be done in linear time. On the other hand, a different alternative is to use the same penalty parameter for all the vectors corresponding to the same level of smoothness. For instance, one can use $\lambda_u, j = \lambda_{u,i}$ if $D^{u_j} = D^{u_i}$. This reduces the burden of cross-validation.

3.6 Convergence analysis

We now examine the convergence of the block-coordinate algorithms developed in the previous section. Here, we assume that $J = 1$ in Model (3.1). In this case we recall that the underlying true tensor can be decomposed as the outer product of vectors $u^* \in \mathbb{R}^L$, $v^* \in \mathbb{R}^T$ and $w^* \in \mathbb{R}^S$, times a constant d^*. Moreover, we assume that the matrices D are chosen to be either fused lasso or trend filtering penalties. Thus, $D^u = D^{(k_u+1)} \in \mathbb{R}^{(L-k_u)\times L}$, $D^v = D^{(k_v+1)} \in \mathbb{R}^{(T-k_v)\times T}$ and $D^w = D^{(k_w+1)} \in \mathbb{R}^{(S-k_w)\times S}$ with k_u, k_v and $k_w \in \{0, 1\}$.

Our proof is inspired by the work on convergence rates for generalized lasso regression problems from [149]. The theorem states that, when starting with good initials, it is necessary to sweep through the data only once. The proof of the claim is based on the identity

$$\mathrm{P}(A \cap B \cap C) = \mathrm{P}(A)\mathrm{P}(B \mid A)\mathrm{P}(C \mid A \cap B),$$

for any events A, B and C. A related statement can be made in the case of multiple factors for a single update depending on the other factors. See the result in the supplementary material; the main difference there involves an error measurement that depends on the factors taken as fixed.

Theorem 3.6.1. *Let $\{u^1, v^1, w^1\}$ denote a one-step update from Algorithm 1, based on initial values $\{u^0, v^0, w^0\}$, and assume that $\|D^u u^*\|_1 \leq c_w$, $\|D^v v^*\|_1 \leq c_w$, and $\|D^w w^*\|_1 \leq c_w$. Then, there exists a constant $c > 0$ such that if $t > 0$ satisfies*

$$\max\left\{\tfrac{4\,t}{d^*} + \tfrac{c\,c_u\,L^{k_u+1/2}\sqrt{\log L}}{d^*}, \tfrac{4\,t}{d^*} + \tfrac{c\,c_v\,T^{k_v+1/2}\sqrt{\log T}}{d^*}, \tfrac{4\,t}{d^*} + \tfrac{c\,c_w\,S^{k_w+1/2}\sqrt{\log S}}{d^*}\right\} \leq \tfrac{1}{2^5},$$

and $\|v^0 - v^*\|_2 < 2^{-1/2}$, $\|w^0 - w^*\|_2 < 2^{-1/2}$, then

$$\begin{aligned} &P\Big(\|u^1 - u^*\|_2^2 \leq 16\left(\tfrac{4\,t}{d^*} + \tfrac{c\,c_u\,L^{k_u+1/2}\sqrt{\log L}}{d^*}\right), \|v^1 - v^*\|_2^2 \leq \\ &\quad 16\left(\tfrac{4\,t}{d^*} + \tfrac{c\,c_v\,T^{k_v+1/2}\sqrt{\log T}}{d^*}\right), \|w^1 - w^*\|_2^2 \leq 16\left(\tfrac{4\,t}{d^*} + \tfrac{c\,c_w\,S^{k_w+1/2}\sqrt{\log S}}{d^*}\right)\Big) \\ &\geq \Psi(t, L)\,\Psi(t, T)\,\Psi(t, S), \end{aligned}$$

where

$$\Psi(t, x) = \left(1 - \sqrt{\frac{2}{\pi}}\frac{1}{t}e^{-\frac{t^2}{2}} - \frac{2^{1/2}}{x^{3/2}\sqrt{5\,\pi\,\log(x)}}\right).$$

Proof. See Appendix B.3.2. □

Theorem 3.6.1 states that with good initials our rank-1 decomposition algorithm will be very close to the true factors under weak assumptions concerning the smoothness of the true factors. Thus, in practice before running our algorithms, we can consider a simple initialization that consists of solving Algorithm (1) for the case where the matrices D^u, D^v and

D^w are all zero. This is known as the power method [86]. Moreover, statistical guarantees for a closely related method to this procedure were studied in [4].

Finally, it should be note that Theorem 3.6.1 implicitly suggests that an appropriate choice of tuning parameter is

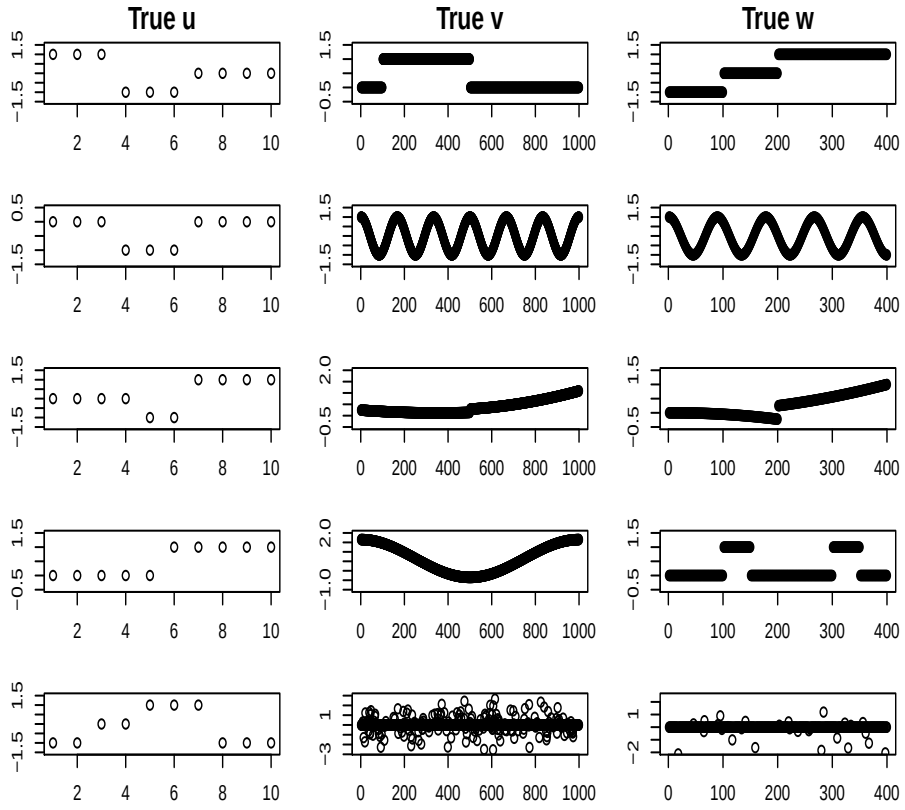
$$(c_u, c_v, c_w) = (\|D^u u^*\|_1, \|D^v v^*\|_1, \|D^w w^*\|_1),$$

which only involves the true latent vectors. In the case of the unconstrained version, a very similar statement to Theorem 3.6.1 holds by taking $\lambda_u = O(L^{k_u+1/2}\sqrt{\log(L)})$, $\lambda_u = O(T^{k_v+1/2}\sqrt{\log(T)})$ and $\lambda_w = O(S^{k_w+1/2}\sqrt{\log(S)})$.

3.7 Experiments

Our experiments focus mainly on the task of rank-1 recovery, since all of our algorithms are based on the development of a rank-1 PTD. For all our simulations we use the Frobenius norm of the difference between the estimated and true tensors as a measure of overall accuracy. The Frobenius norm is a natural choice of model fit, since we also benchmark against a recovery method that does not directly produce a rank-1 tensor but does provide an estimate of the true mean tensor. This method is based on the idea of stacking several penalized matrix decompositions using the technique from [151]. Specifically, we consider the tensor of observations $\underline{\tilde{X}}$ as a collection of 10 distinct 1000×400 matrices, each of which is estimated via a rank-1 PMD. This will lead to 10 estimated rank-1 matrices which are

Figure 3.2: Each row gives rise to a different structure by taking the outer product on the corresponding, horizontally plotted, vectors.



concatenated to build a $10 \times 1000 \times 400$ tensor. We call this procedure, with an abuse of notation, PMD(P_v, P_w) where P_v and P_w are the penalties on v and w, when computing the rank-1 PMD matrices. The other methods included in the study are the PTD with different penalties P_u, P_v, P_w, denoted as PTD(P_u, P_v, P_w). We consider choices such as the L1 penalty, the fussed lasso (FL) and trend filtering of order k (TFk). Note that we are implicitly

comparing to the method from [4] since, for rank-1 recovery, this reduces to the power method, and hence to PTD(L1,L1,L1) for appropriate parameters.

For our simulations, the tuning parameters are chosen by cross validation on a grid of possible values for each of the parameters λ_u, λ_v, and λ_w. For every, we randomly select 10% of the data for testing, using the other 90% as training data. Out of a range of candidate tuning parameters we select those that produce the smallest error on the 10% held-out set. This process is repeated for each of 100 simulations, for different methods and structures, in order to obtain average Frobenius errors for all the competing methodologies with respect to every structure.

To see how different choices of penalties can behave under different scenarios, we ran experiments using five different rank-1 tensors as the true mean tensor. These choices are designed to explore a range of plausible structures that we might find in real problems. For the first structure both v and w are piecewise flat. For the second, both v and w are periodic functions. For the third, both v and w are piecewise quadratic polynomials. For the fourth, v is smooth and w is piecewise constant. For the fifth, both v and w are sparse but with no specific structural pattern like smoothness or flatness. The goal of this final scenario is to understand how structural penalties perform in a data set where they are not warranted. Further details of this simulation are included in the supplementary material. Figure 3.2 also shows a plot of these different structures.

The results of our simulation study are shown in Table 3.1. In all

Table 3.1: Comparison of the Frobenius norm error between the true tensor and the estimated tensor using different methods.

Method	Structure 1	Structure 2	Structure 3	Structure 4	Structure 5
PTD(L1,L1,L1)	37.37	47.63	46.16	39.91	40.58
PTD(L1,FL,FL)	6.31	27.54	11.76	10.30	57.15
PTD(L1,TF1,FL)	15.07	20.49	11.55	9.00	70.32
PTD(L1,TF1,TF1)	17.61	14.40	11.85	12.40	79.25
PMD(L1,L1)	85.05	89.10	100.70	91.89	72.87
PMD(L1,FL)	49.09	50.14	52.70	22.73	92.20
PMD(FL,FL)	15.05	43.17	25.64	33.95	114.09

cases, PTD converged with few iterations, usually less than 10. From these results, it is clear that different choices of penalty are suitable for different problems. For structure 1, in which the true v and w are piecewise flat, the combination PTD(L1, FL, FL) outperforms all the other choices that we considered. Interestingly, PTD(L1,TF1,FL) and PTD(L1, TF1,TF1) provided better results than the "stacking" method PMD(FL,FL). Note also that PTD(L1,TF1,FL) and PTD(L1,TF1,TF1) behave fairly similar to one another. This is expected since a piecewise constant function is a special case of a piecewise linear function and hence we would expect that TF1 would produce only slightly worse results than fused lasso.

Moreover, Table 3.1 also illustrates when our methodology should not be expected to work. This is what happens with structure 5, where there is no spatial pattern in the true vectors u, v and w, and instead they are merely sparse (80% of their coordinates are zero). Here, as expected, PTD(L1,L1,L1) outperforms any of our methods.

Table 3.2: Comparison of the Frobenius norm error between the true tensor and the estimated tensor using for different levels of noise and a fixed structure, averaging over 100 Monte Carlo simulations

Method	$\sigma = 1.25$	$\sigma = 1.50$	$\sigma = 1.75$	$\sigma = 2.00$	$\sigma = 2.25$
PTD(L1,L1,L1)	62.66	81.66	80.46	99.50	94.37
PTD(L1,FL,FL)	32.61	38.80	41.63	46.32	49.33
PTD(L1,TF1,FL)	24.55	28.55	32.35	37.87	38.43
PTD(L1,TF1,TF1)	17.00	21.35	22.27	27.09	27.36
PMD(L1,L1)	116.19	139.57	158.71	185.05	209.45
PMD(L1,FL)	66.80	76.81	83.65	98.18	111.09
PMD(FL,FL)	52.43	57.52	65.36	83.98	92.71

In the previous experiment we simulated all data sets with the assumption that the noise had variance 1. Now we fix the rank-1 tensor mean of Structure 2, where both v and w are periodic functions, and then we compare the performance of different methods as the standard deviation of the noise changes. Recalling that in Structure 2 both v and w are periodic smooth, it does not come as a surprise that PTD(L1,TF1,TF1) provides the best performance in all situations considered in Table 3.2. In addition, it is clear that the error of all methods increases as the variance of the noise does. Nevertheless, the performance of our method seems to be the most stable.

Finally, we evaluate the recovery of mean tensors having multiple factors, with $\sigma = 1$. Scenarios where the true model consists of $J = 2$ and $J = 3$ are considered. Our comparisons are based on taking sums of different rank-1 tensors using the structures discussed before. The competing methods are PTD(L1,FL,FL) and PTD(L1,TF1,TF1), versus Algorithm 1 from

Table 3.3: Comparison of the Frobenius norm error between the true tensor and the estimated tensor using different methods, averaging over 100 Monte Carlo simulations

Method	Structures									
	1,2	1,3	1,4	2,3	2,4	3,4	1,2,3	1,2,4	1,3,4	2,3,4
Anandkumar	544.0	310.5	85.4	121.0	128.9	273.4	534.9	555.3	346.6	350.3
PTD(L1,FL,FL)	55.3	46.5	27.1	71.2	59.7	107.3	184.2	48.3	102.8	126.1
PTD(L1,TF1,TF1)	51.7	71.6	67.8	49.2	50.9	94.0	120.3	75.2	141.6	120.8

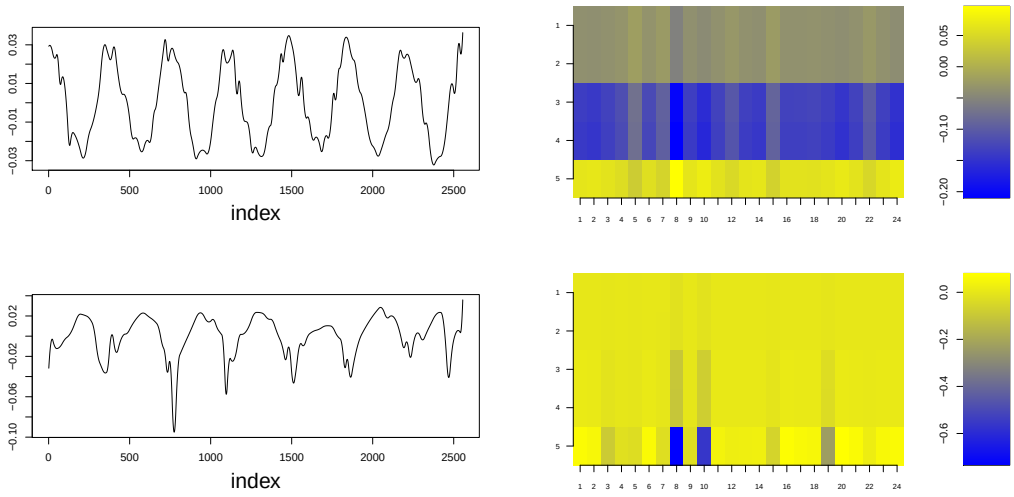
[4]. For the latter, we set the number of initializations $L = 30$ and the number of iterations $N = 10$. The results in Table 3.3 show a clear gain for our approach over the method from [4], which does not impose any smoothness constraints on its solutions.

3.8 Real data examples

3.8.1 Flu hospitalizations in Texas

As a simple illustrative example, we consider measurements of flu activity and atmospheric conditions in Texas; see the supplementary material for information how to collect the data. There are 5 variables measured daily across 24 cities in Texas from January 1, 2003 to December 31, 2009. The variables are: maximum and daily average observed concentration of particulate matter (air quality measure), maximum and minimum temperature, and a measure of flu intensity capturing flu-related hospitalizations per million people. The data tensor is thus a 5x24x2556 array where we expect clear temporal patterns, along with correlations among the five vari-

Figure 3.3: (a) Time vector for the first factor (b) Loadings matrix for first factor (c) Time vector for second factor (d) Loadings matrix for second factor.



ables. For example, during the winter months we would expect an increase in flu-related hospitalizations, correlated with seasonal patterns of maximum and minimum daily temperatures.

To show the kind of interesting results that one can get with our methods, we compute a two-factor Parafac decomposition. We use trend filtering of order 2 in the temporal mode and no penalty on the other two modes (although it would be straightforward to incorporate a penalty on the spatial mode as well.) We use our main result (3.5.1) to find the factors using coordinate-wise optimization. The tuning parameter for the trend-filtering penalty is chosen by cross validation from a grid of values to ensure that we get a smooth vector for the time mode.

We considered fitting models with different values of J, we found that a model with one factor explains 36% of the variance, a model with two factors explains 45% percent of the variance, and a model with three factors results in increase in variance explained of less than 1% with respect to the case $J = 2$. Moreover, the model with 3 factors results in highly correlated factors. For this reason we use a model with 2 factors.

From Figure 3.3 we note a clear seasonal effect. In the first factor we observed that the loadings for the flu intensity, minimum temperature, and maximum temperature can be all explained in a similar way. For the first of these three variable the loadings are all positive. Hence, given the shape of the time vector we see a periodic pattern of flu cases across cities with the highest during the winter months and the lowest during the summer months.

3.8.2 Motion capture data

For a more challenging task, we evaluate the performance of our PTD method using data from the motion capture (moCap) repository at `mocap.cs.cmu.edu`. This consists of subjects performing different physical activities in repeated independent trials. We construct 3-array tensors by taking sets of videos as one mode, 12 representative variables of the body movements as the second mode, and data frames in time as the third mode. The 12 variables are listed in the supplementary material.

We built 2 tensors each for 5 different tasks, with each task generat-

ing a training-set tensor and a test-set tensor. The training set tensor corresponds to a single subject performing multiple repetitions of a single related set of physical activities. Similarly, the corresponding test-set tensor corresponds to that same subject performing further repetitions of those same activities. For example, the first data set (comprising 1 tensor in the training set and 1 tensor in the test set) is called 126-swimming; this is formed by looking at 8 videos of subject 126 performing different swimming styles. In the moCap repository, videos 1,3,6,8 are used for training while videos 2,4,7,9 are used for testing. This results in both tensors having dimensions $4\times253\times12$.

The other four data sets, explained in detail in the supplementary material, are 138-story (subject 138 walking and moving arms); 107-walking (subject 107 walking with obstacles); 9-running (subject 9 running); and 138-marching (just like it sounds). For these data sets, the tensors dimensions are $4\times325\times12$, $4\times828\times12$, $4\times128\times12$, $4\times371\times12$, respectively.

In this context, our PTD approach can be thought of as a smoothing step applied to the training-set tensor, to yield better out-of-sample predictions for the test-set tensor. We evaluate the performance of the method by calculating the reconstruction error (again, by Frobenius norm) when using the fitted/smoothed training-set tensor to predict the corresponding test-set tensor.

We find that for the tensors considered here, rank-1 is the best Parafac decomposition, since models with higher factors result in strongly corre-

Table 3.4: Comparison of the Frobenius norm error between the estimated tensor and the test tensor for the moCap datasets

Method	Task				
	126-swimming	138-story	107-walking	9-running	138-marching
Anandkumar	254.80	134.63	135.17	84.40	143.86
PTD(L1,TF2,TF2)	250.98	131.78	134.92	84.29	142.44
PMD(L1,TF2,TF2)	267.89	145.14	143.43	88.06	149.41

lated factors. We ran our rank-1 PTD with a trend-filtering penalty of order 2 on the second mode, and no constraints in the other modes. We compare against the PMD using the same degree of smoothness, as well as the classical PCA method from [4]. From Table 3.4 it is clear that PTD offers the best performance. Thus we can see the gain of using smooth penalties, reflecting the fact that physical movements involve motion-capture variables that change smoothly in time. Moreover, it is clearly favorable to pool information across videos, as our method does, rather than treating them independently, as with the PMD algorithm.

3.9 Discussion

In many problems, tensors offer a natural way to represent high-dimen- sional, multiway data sets. However, tensors by themselves are difficult to interpret, creating the need for methods that shrink towards some simpler, low-dimensional structure.

Parafac models have been widely used for this task, but existing

state-of-the-art methods typically constrain the factors to be orthogonal, or simply do not enforce any constraints. As we have shown, this can be undesirable in practice, especially if one is looking for more interpretable factors, where there is a natural spatial or temporal relation between observation, and it is expected that the factors will be smooth. We fill this gap by providing a set of methods that offer piecewise smooth Parafac decompositions. Our methods exploit state–of–the–art convex optimization algorithms and are shown to have excellent performance in our experiments.

Finally, we have shown two alternatives for finding our smooth tensor decompositions with generalized lasso penalties. The constrained formulation seems to be an attractive option for practitioners, with clear intuitive control over the level of smoothness exhibited by the solutions. On the other hand, in light of its computational advantages, the unconstrained formulation offers a more practical approach, especially if there is no pre-existing knowledge about the anticipated smoothness of the solutions.

Chapter 4

Nonparametric density estimation by histogram trend filtering

4.1 Nonparametric density estimation

This chapter is devoted to the topic of density esimation, the work is based on the working paper [98].

Consider the classic estimation problem in $\mathbb{R}^d$, where we observe $y_i \sim f_0$ for $i = 1, \ldots, n$ and wish to estimate f_0. Most data-analysis practitioners that confront this problem turn to kernel density estimation, due to its familiarity, its computational efficiency, and its well-understood statistical properties. Yet kernel methods are known to suffer from the local-adaptivity problem, wherein the used of a fixed bandwidth parameter may result in simultaneously undersmoothing and oversmoothing in different regions of the density.

A huge variety of methods have been proposed that improve upon basic kernel methods in a way that addresses this problem, from adaptive kernel bandwidths to penalized-likelihood estimation. Yet these methods typically either incur a much higher computational burden than basic kernel methods, or else they involve hyperparameters that are difficult to spec-

ify and tune. The goal of this chapter is to address this gap. We propose a method called histogram trend filtering, which solves the adaptivity problem while simultaneously satisfying the following criteria:

1. It is computationally efficient, even for large data sets.

2. It has strong statistical guarantees.

These two factors make our proposed method a strong candidate to replace ordinary kernel density estimation as the default "first pass" for data-analysis practitioners.

In the real line, the histogram trend-filtering estimator is related to the following variational optimization problem based on penalizing the log likelihood $g = \log f$:

$$\begin{aligned} \underset{g}{\text{minimize}} \quad & -\sum_{i=1}^{n} g(y_i) \\ \text{subject to} \quad & \int_{\mathbb{R}} e^g = 1 \\ & J(g) \leq t \,, \end{aligned} \tag{4.1}$$

where $J(g)$ is a known penalty functional. Imposing an appropriate penalty can encourage smoothness and avoids estimates that are sums of point masses.

Specifically, we consider solutions to (4.1) for penalties based on total variation, as proposed by [83]. We provide conditions under which explicit rates of convergence can be obtained for these estimators. We also study a finite-dimensional version of this variational problem—histogram trend

filtering—which involves two conceptually simple steps. First, partition the observations into D_n histogram bins with centers $\xi_1 < \cdots < \xi_{D_n}$ and counts $x_1, \ldots, x_{D_n}$. Then assume the surrogate model $x_j \sim \text{Poisson}(\lambda_j)$ and estimate the λ_j's via polynomial trend filtering [79, 140] applied to the Poisson likelihood. The renormalized λ_j's then may be used to form an estimate of f_0.

Our results show that this simple, computationally efficient procedure yields excellent performance for density estimation. Our main theorem characterizes how the optimal bin size must shrink as a function of n to ensure consistency for estimating f_0, and provide bounds on the proposed procedure's reconstruction error under the assumption that the bins are chosen accordingly. Our empirical results also show that the histogram trend-filtering estimator is adaptive to changes in smoothness of the underlying density when familiar information criteria are used to choose the method's single tuning parameter. Put simply, it can yield an estimate that is simultaneously smooth in some regions and spiky in others. This behavior contrasts favorably with kernel density estimation, where the bandwidth parameter governs the global smoothness of the estimate.

4.2 Histogram trend filtering in one dimension

The idea of histogram trend filtering is to reduce the density estimation problem to that of a nonparametric Poisson regression problem, which is solved by trend filtering [79, 141]. The method is so computationally

efficient for two reasons: (1) because binning the data results in a huge reduction from data points to bin counts, and (2) because the trend-filtering estimator for a Poisson regression can be obtained so cheaply, using the extraordinarily fast ADMM algorithm of [110]. An important point for us to demonstrate is that the data reduction step can be done without losing too much information; we address this concern later.

Let us now construct in detail the histogram trend-filtering estimator for one-dimensional problems, which can be viewed as a discrete approximation to Problem (5.3) when J penalizes the total variation of g or higher-order versions thereof. We begin with several assumptions made for ease of exposition. Let $\mathcal{X} \subset \mathbb{R}$ denote the support of f_0. Suppose that $\mathcal{X}$ is a compact set that it is partitioned into D_n disjoint intervals I_j with midpoints ξ_j, such that $\bigcup_j I_j = \mathcal{X}$. We assume that the intervals are of equal length δ_n and ordered so that $\xi_1 < \cdots < \xi_{D_n}$. Any of these assumptions can be relaxed in practice.

Now consider a histogram of the observations using bins I_j. Let $x_j = \#\{y_i \in I_j\}$ denote the histogram count for bin j, and consider the surrogate model

$$x_j \sim \text{Poisson}(\lambda_j) \,, \quad \lambda_j = n\delta_n f_0(\xi_j) \approx n \int_{I_j} f_0(y)\, dy \,. \tag{4.2}$$

Let $\theta_j = \log \lambda_j$ be the log rate parameter for bin j, let $\theta = (\theta_1, \ldots, \theta_{D_n})$, and define the loss function

$$l(\theta) = \sum_{j=1}^{D_n} \left\{ e^{\theta_j} - x_j \theta_j \right\} ,$$

as the negative log likelihood corresponding to Model (4.2). We propose to estimate θ using the solution to the unconstrained optimization problem

$$\underset{\theta \in \mathbb{R}^D}{\text{minimize}} \quad l(\theta) + \tau \|\Delta^{(k+1)}\theta\|_q^p \,, \tag{4.3}$$

where $\Delta^{(k+1)}$ is the discrete difference operator of order k. Concretely, when $k = 0$, $\Delta^{(1)}$ is the matrix encoding the first differences of adjacent values:

$$\Delta^{(1)} = \begin{pmatrix} 1 & -1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & -1 & 0 & \cdots & 0 \\ \vdots & & & & \ddots & \vdots \\ 0 & \cdots & & 0 & 1 & -1 \end{pmatrix}. \tag{4.4}$$

For $k \geq 1$ this matrix is defined recursively as $\Delta^{(k+1)} = \Delta^{(1)}\Delta^{(k)}$, where $\Delta^{(1)}$ from (5.9) is of the appropriate dimension.

We focus on problem (4.3) when $q = p = 1$, which corresponds to the polynomial trend-filtering estimator under a Poisson likelihood. Intuitively, the trend-filtering estimator is similar to a locally adaptive spline model: it places a lasso penalty on a discrete analogue of the order-k derivative of the underlying log-density, resulting in a piecewise polynomial estimate whose degree depends on k. From the point of view of splines, the trend-filtering penalty appeared in the work of [102] for the context of nonparametric regression. With the language just used above, trend filtering has been studied extensively in the context of function estimation, generalized linear models, and graph denoising [79, 141, 149, 100].

From a computational perspective, we emphasize that in the case of interest here, (4.3) with $q = p = 1$, the histogram trend filtering estimator

can be found efficiently. If $k = 0$, for a fixed value of the tuning parameter the estimator can be found with $O(D_n)$ cost, see [74]. In the more general case, $k > 0$, (4.3) with $q = p = 1$ can be efficiently solved using the recent developments in ADMM algorithms for trend filtering problems [110]. Such approach is based on introducing a convenient slack variable that then leads to an ADMM in which each update can be done with $O(D_n)$ cost.

4.3 Previous work

4.3.1 Other adaptive and penalized likelihood density estimators

In this section we present a brief review of density estimators related to our methods. We begin by discussing the seminal work from [65] which can be motivated from a Bayesian perspective. This starts by considering the prior

$$p(f) \propto \exp\left(-\Phi(f)\right) \mathbb{I}\left(f \in \mathcal{A}\right),$$

where Φ is a roughness penalty and $\mathcal{A}$ is some class of density functions. Then, given the usual likelihood

$$p(y \mid f) = \prod_{i=1}^{n} f(y_i),$$

the authors in [65] produce a maximum a posteriori (MAP) estimate of f_0 by solving

$$\hat{f} = \text{argmin}_{f \in \mathcal{A}} - \log p(y \mid f) + \Phi(f)\,. \tag{4.5}$$

This is the main focus of study in [65], where different roughness penalties were considered.

In a variation of the estimator from [65], [127] works within the framework of penalized likelihood. However, rather than penalties directly imposing constraints on the density space, [127] proposes to penalize the log density. This is immediately attractive since it automatically imposes a positive constraint in the estimates, with the formal formulation given as

$$\begin{array}{ll} \underset{g}{\text{minimize}} & -\frac{1}{n}\sum_{i=1}^{n} g(y_i) + \frac{1}{2}\tau\,\Phi(g) \\ \text{subject to} & \int e^{g(\mu)} d\mu = 1. \end{array} \tag{4.6}$$

The roughness penalties studied in [127] are of the form

$$\Phi(g) = \int_0^1 \left[D(g)(\mu)\right]^2 \, d(\mu),$$

where $D(g)$ is a function of the first m derivatives of g, see [127] for the specific construction. There, Theorem 3.1 also shows that (4.6) is equivalent to the unconstrained problem

$$\underset{g}{\text{minimize}} \; -\frac{1}{n}\sum_{i=1}^{n} g(y_i) + \frac{1}{2}\tau\,\Phi(g) + \int e^{g(\mu)} d\mu. \tag{4.7}$$

This alternative formulation has the nice feature that can be formulated as a convex optimization problem, see [108].

It turns out that a similar result can easily be proven for our Poisson surrogate problem. This is given in the following Theorem.

Theorem 4.3.1. *With the notation from Section 4.2, it can be proven that there exists a constant $c > 0$ such that $\hat{\theta}$ solves (4.3) if and only if $\hat{g}_i = \hat{\theta}_i - \log(n\,\delta_n)$ solves*

$$\begin{array}{ll} \underset{g}{\textit{minimize}} & -\frac{1}{n}\sum_{i=1}^{D_n} x_i\, g_i \\ \textit{subject to} & \sum_{i=1}^{D_n} \delta_n\, e^{g_i} = 1, \quad \|\Delta^{(k+1)} g\|_q^p \leq c. \end{array} \tag{4.8}$$

Proof. See Appendix C.1. □

Thus, we have shown that our Poisson surrogate problem is indeed a discretization of problem (5.3), where we replace the classical likelihood by a cross entropy objective, the integrability constraint by a constraint on the midpoint rule for the estimator, and the total variation penalty by a discrete version using difference matrices.

While our histogram trend filtering approach to density estimation might seem closely related to the estimator from [127], there are two significant differences. First, as pointed out by [120], the estimator given by problem (4.6) tends to over-smooth, since non-smoothness is penalized more heavily at low density values than at high density values, which may lead to estimation problems. The authors in [120] address this problem by imposing a a total variation penalty. Thus, giving rise to the problem

$$\begin{array}{ll} \underset{f}{\text{minimize}} & -\frac{1}{n}\sum_{i=1}^{n}\log(f_i) + \tau\sum_{i=2}^{n}|f_i - f_{i-1}| \\ \text{subject to} & a^T f = 1, \end{array} \tag{4.9}$$

for some integration coefficient vector a and parameter $\tau > 0$. The main motivation for this problem is to avoid the over-smooth solutions from solving (4.6). Hence, given the flexibility of imposing $\|\cdot\|_q^p$, our histogram trend filtering estimators are also expected to avoid over-smoothing by taking $p = q = 1$. However, the other important issue associated with the estimator given by (4.6) is the computational complexity. This is also shared by the estimator from [120] since both of these procedures require to estimate a

vector in R^n. In contrast, we solve optimization problems in a significantly lower dimensional space, R^{D_n}.

Next we observe that by its mere definition in Problem 4.3, when $p = q = 1$ our density estimator provides piecewise polynomial solutions in the log-space. Here, the parameter k in the difference matrix indicates the degree of the polynomial approximation used. For instance, $k = 0$ corresponds to piecewise constant solutions, while $k = 1$ to piecewise linear solutions. An attractive feature of our method is that it is not necessary to specify the the locations of break points; this is done adaptively by solving a convex optimization problem. In contrast, [9] considered fitting splines in the log-space but this requires specification of the locations of the of the knots. Moreover, [9] provides rates of convergence for such spline estimators, in terms of the Kullback-Leiber divergence, when the true density satisfies

$$\int | \left(\log f_0\right)^{(k+1)} |^2 < \infty, \tag{4.10}$$

with the superscript $(k+1)$ denoting the $(k+1)$-the derivative.

Next, we review the penalized estimator from [150]. This is obtained by solving the problem

$$\begin{array}{rcll} \hat{f}_W & = & \underset{f}{\arg\min} & -\frac{1}{n}\sum_{i=1}^n \log(f(y_i)) + \text{pen}(f) \\ & & \text{subject to} & \int f = 1,\ f \in \mathcal{C} \end{array} \tag{4.11}$$

where $\mathcal{C}$ is a class of non-negative piecewise polynomials and $\text{pen}(f)$ is a functional that penalizes the complexity of polynomials. The solution to

(4.11) enjoys attractive theoretical properties, and [150] shows that if f_0 is a member of the Besov space $B_q^\alpha\left(L_p([0,1])\right)$ where $\alpha > 0$, $1/p = \alpha + 1/2$ and $0 < p < q$, then,

$$E\left[\|f_0^{1/2} - \hat{f}_W^{1/2}\|_2^2\right] \leq C\left(\frac{\log_2^2(n)}{n}\right)^{\frac{2\alpha}{2\alpha+1}}.$$

Moreover, the estimator $\hat{f}_W$ involves using recursive dyadic partitions in order to produce near-optimal, piecewise polynomial estimates, analogous to the methodologies in [16, 85] and [44]. Also, this multiscale method provides spatial adaptivity similar to wavelet-based techniques [46, 77], with a notable advantage. Wavelet-based estimators can only adapt to a functions smoothness up to the wavelets number of vanishing moments; thus, some a priori notion of the smoothness of the true density or intensity is required in order to choose a suitable wavelet basis and guarantee optimal rates. The estimator $\hat{f}_W$, in contrast, automatically adapts to arbitrary degrees of the functions smoothness without any user input or prior information. However, this penalized method requires elevated computational effort. Specifically, it it involves $O(n\ \log_2(n))$ calls to a convex minimization routine and $O\left(n\ \log_2(n)\right)$ comparisons of the resulting (penalized) likelihood values.

4.3.2 Log-Density estimation by total variation

Finally, we recall the estimator from [82] which solves a discrete variant of (5.3). This is written as

$$\underset{g \in \mathbb{R}^n}{\text{minimize}} \quad -\frac{1}{n} \sum_{i=1}^{n} g_i \; + \; \sum_{i=1}^{m} c_i \; e^{g_i} + \tau \, \|D \, g\|_1, \tag{4.12}$$

where the constants $\{c_i\}$ are used to encode the integrability constraint, and D is matrix that encodes the smoothness of the desired solution. We notice that (4.12) looks similar to our formulation in (4.3), however there are some important differences. First, rather than working with the full likelihood as (4.12), we focus on weighted likelihood which as discussed earlier reduces the computation complexity significantly. Interestingly, [82] did briefly mentioned a weighted version of the problem though not based in binning the observations. Moreover, [82] proposed to handle the non-differentiable penalty $\|\cdot\|_1$ using a combination of two differentiable constraints. This makes their problem hard to solve hence having to rely on general software for convex optimization. In contrast, we have shown that our formulation is still amenable for "Silverman's" trick which allows us to put the integrability constraint as part of the objective. Thanks to the latter, we were then able to propose the fast algorithm from [110] which exploits the structure of the problem.

4.3.3 Lindsey's method

One of the key aspects behind our histogram trend filtering procedure is the Poisson surrogate model (or weighed likelihood in the optimization problem) implied by binning the data. Here, we emphasize that this "binning" idea is actually quite old in the statistics literature. Originally, the Poisson approximation to the data appeared in [94, 95]. This was then thoroughly discussed for density estimation problems in [53]. More recently, [17] considered a variant of the Poisson surrogate model (4.2). The authors perform the transformation

$$\tilde{x}_j \;=\; \sqrt{x_j + \frac{1}{4}}, \;\; j = 1, \ldots, D_n,$$

and show that the variance of each $\tilde{x}_j$ is roughly constant. Hence, [17] argues that it is attractive to fit a non-parametric regression model to the observations $\{\tilde{x}_j\}_{j=1}^{D_n}$, since it is also true that $E(\tilde{x}_j) \approx \sqrt{\lambda_j}$. The resulting method from [17] proceeds by choosing the wavelet block thresholding for estimating $\{\sqrt{\lambda_j}\}_{j=1}^{D_n}$ (for a description of the general block thresholding method see [19]). After $\{\sqrt{\lambda_j}_{j=1}^{D_n}\}$ have been estimated, one can square such estimates and rescale them in order to get an estimate of truth density at each bin.

4.4 Statistical convergence

Next we focus on the version of the estimator where we approximate the function on the discrete grid. This is necessary for finding the solution

by numerical optimization, and is analogous to the approach taken by [83], who started with a variational problem and then moved to an approximate solution on a grid. However, they did not provide any statistical guarantees for either of their formulations.

We now denote the regularization parameter as τ_n and define the vectors

$$\theta^0 := \{\log n - \log D_n + \log f_0(\xi_1), \ldots, \log n - \log D_n + \log f_0(\xi_{D_n})\}, \quad (4.13)$$

and $\hat{\theta}$ as the solution to Problem (4.3). Thus up to a known constant of proportionality, θ^0 and $\hat{\theta}$ are the true and estimated log densities, respectively.

Our next theorem provides a bound on estimation error that refers to the penalty explicitly. The state result can also be extended to densities of unbounded support.

Theorem 4.4.1. *Let ξ_j' be the point in I_j satisfying*

$$\delta_n \, f_0(\xi_j') = \int_{I_j} f_0(t)dt.$$

Let us also take $b \in (0, 1/2)$ and define $\hat{\theta}$ as the solution to the convex optimization problem

$$\begin{array}{ll} \underset{\theta}{\textit{minimize}} & \sum_{j=1}^{D_n} \left\{ e^{\theta_j} - x_j \theta_j \right\} + \tau \left\| \Delta^{(k+1)} \theta \right\|_1 \\ \textit{subject to} & |\theta_j - log(n\,\delta_n)| \leq n^b, \;\; j = 1, \ldots, D_n. \end{array} \quad (4.14)$$

Let us assume that $(f_0(\xi_j'), \ldots, f_0(\xi_{D_n}'))$ belongs to the constraint set of (4.14). Then

$$\hat{f}(\xi_j') = \frac{\exp\left(\hat{\theta}_j\right)}{n\,\delta_n}, \;\; j = 1, \ldots, D_n,$$

satisfies

$$\sum_{j=1}^{D_n} \delta_n \, f_0(\xi_j') \log\left(\frac{f_0(\xi_j')}{\hat{f}(\xi_j')}\right) = O_P\left(\frac{\| \left(\Delta^{(k+1)}\right)^- \|_\infty}{D_n^r} \|\Delta^{(k+1)} \log\left(f_0(\xi')\right) \|_1 + \frac{1}{n^{r/s-b}}\right), \tag{4.15}$$

where we choose $D_n = \Theta\left(n^{1/s}\right)$ *and* $\tau = \Theta\left(n^{1-r/s} \| \left(\Delta^{(k+1)}\right)^- \|_\infty\right)$*, where* $s > 1$ *and* $r \in (0, s/2)$*.*

Proof. See appendix C.1.1. □

Theorem 4.4.1 states convergence rates for our Poisson surrogate model estimator. In particular, the bound in (4.15) controls the Kullback-Leibler divergence between our estimator and a discretized version of the true density. Moreover, the constraint on the supremum norm in the log-space space ensures that the optimization is over a compact set. Hence, it is not restrictive given that this bound tends to infinity as n increases.

Finally, we emphasize that $\| \left(\Delta^{(k+1)}\right)^- \|_\infty = O(D_n)$. This is a consequence of the proof of Corollary 4 in [149]. In practice, we have found that $\| \left(\Delta^{(k+1)}\right)^- \|_\infty D_n^{-1} \in (.1474, .1482)$ if if D_n is chosen between 500 and 10000.

4.5 Model selection

We know turn to the discussion of the parameters D_n and τ when $p = q = 1$. For the former of these parameters, we see from the results in the previous section that $D_n = O(n^{1/s})$, for $s > 2$, seems a reasonable choice. In our experience, the rule $D_n = 10\, n^{1/2.5}$ performs excellently. On the other

hand, for the choice of τ, we see from Theorem 4.4.1 that

$$\tau = \Theta\left(n^{1-r/s} \| \left(\Delta^{(k+1)}\right)^{-} \|_{\infty}\right),$$

is a candidate choice with r satisfying the constraint from Theorem 4.4.1. In particular, this choice ensures consistency for the trivial case in which the true density f_0 is uniform, the precise definition of the universal penalty required in [120].

Finally, on the choice of τ, we also consider an add-hoc rule inspired by the work of [142] on regression problems with generalized lasso penalties. This consists of computing the solution path of the problem 4.3 and then considering a surrogate AIC approach by computing

$$\text{AIC}_{\tau} = l(\hat{\theta}_{\tau}) + k + 1 + \left|\left\{i : (\Delta^{(k+1)}\hat{\theta}_{\tau})_i \neq 0\right\}\right|.$$

The parameter τ is then chosen to minimize the expression above.

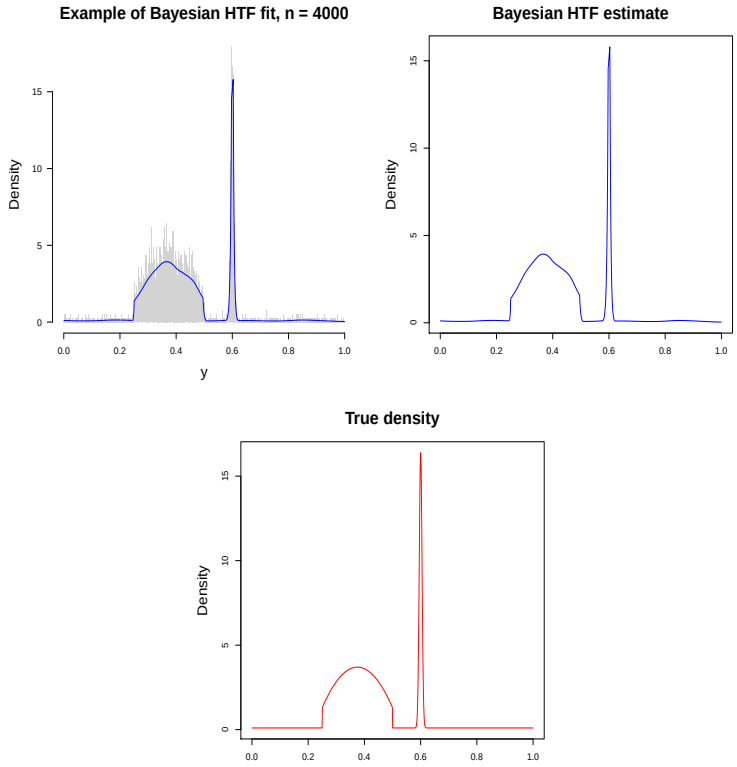
4.6 Bayesian histogram trend filtering

In the previous section we discussed a very natural alternative for model selection. Here, we present a Bayesian perspective of our histogram trend filtering method which is amenable to standard Bayesian model selection criteria.

We write the probability model

$$x_j \quad \sim \quad \text{Poisson}\left(e^{\theta_j}\right), \quad j = 1, \ldots, D,$$

Figure 4.1: Left panel depicts the estimated density provided by our Bayesian HTF, with τ selected using DIC, on top of data generated using the density on the third panel. For purposes of comparison, we also display the Bayesian HTF by itself in the second panel. In this example $n = 4000$.



and place the prior

$$P(\theta) \propto \exp\left(-\tau \left(\theta \, \Delta^{(k+1)}\right)^T \Omega \, \Delta^{(k+1)} \theta\right),$$

where Ω is a diagonal matrix in whose diagonal entries we place independent gamma priors. We notice that while the prior is improper, as the matrix $\Delta^{(k+1)})^T \, \Omega \, \Delta^{(k+1)}$ is not full rank, the posterior is well defined since the log

likelihood,

$$\log P(x|\theta) \;=\; \sum_{j=1}^{D_n} \left(e^{\theta_j} - \theta_j\, x_j\right),$$

is a convex function of θ. Using this fact, our Bayesian model is fitted thorough standard Hamiltonian MCMC, see [106]. This can be done with multiple choices of τ, after which we chose the model that minimizes the Deviance Criterion Information (DIC) computed as in [61], by defining

$$D(\theta) \;=\; -2\,\log P(x|\theta),$$

and then setting

$$\text{DIC} = D(\hat{\theta}_{\text{bayes}}) \;+ 2\left[E_{\theta\sim P(\theta|x)}(D(\theta)) - D(\hat{\theta}_{\text{bayes}})\right].$$

4.7 Histogram trend filtering for 2D density estimation

In this section we show how our histogram trend filtering estimator can be extended to handle multivariate data. We focus on estimation of densities with support on $[0,1]\times[0,1]$ although our discussion can easily be extended to higher dimensions.

The non-parametric density estimation problem in $[0,1]\times[0,1]$ can be stated as follows. We are given measurements

$$y_i \;\sim^{\text{i.i.d}}\; f_0, \quad \text{support}(f_0) \;\subset\; [0,1]\times[0,1],$$

and the goal is to estimate f_0.

Similarly, to the univariate case, given $D_n \in \{1, \ldots, n\}$ we define

$$x_{j,k} \;=\; \#\left\{ y_i \in \left[\frac{j-1}{D_n}, \frac{j}{D_n}\right] \times \left[\frac{k-1}{D_n}, \frac{k}{D_n}\right] \right\},$$

for $j, k \in \{1, \ldots, D_n\}$. Considering the surrogate model

$$x_{j,k} \;\sim\; \text{Poisson}\left(e^{\theta_{j,k}}\right),$$

we solve the penalized likelihood problem

$$\hat{\theta} \;=\; \arg\min_{\theta} \sum_{j=1}^{D_n} \sum_{k=1}^{D_n} \left(e^{\theta_{j,k}} - x_{j,k}\,\theta_{j,k}\right) \;+\; \tau \|\Delta^{(2d,k+1)}\theta\|_1, \tag{4.16}$$

where $\tau > 0$ is a tunning parameter, and $\Delta^{(2d,k+1)}$ is defined as in [149], proceeding as follows. First, let $\Delta^{(2d,1)}$ be the incidence matrix of the 2D grid graph, G. Thus, the $l-$th row of $\Delta^{(2d,1)}$ corresponds to the $l-$th edge of G given by $e_l = (i, j)$ and $\Delta^{(2d,1)}_{l,i} = 1$, $\Delta^{(2d,1)}_{l,j} = -1$ and $\Delta^{(2d,1)}_{l,m} = 0$ for all $m \neq i, j$. Having constructed $\Delta^{(2d,1)}$, we iteratively define $\Delta^{(2d,k+1)}$ as

$$\Delta^{(2d,k+1)} = \begin{cases} -\left(\Delta^{(2d,1)}\right)^T \Delta^{(2d,k)} & \text{if } k \text{ is odd} \\ \Delta^{(2d,1)}\,\Delta^{(2d,k)} & \text{if } k \text{ is even.} \end{cases}$$

After $\hat{\theta}$ is obtained, we estimate the density at the centers of bins as

$$\hat{f}\left(\frac{j-1/2}{D_n}, \frac{k-1/2}{D_n}\right) \;=\; \frac{e^{\hat{\theta}_{j,k}}\; D_n^2}{\sum_{j'=1}^{D_n} \sum_{k'=1}^{D_n} e^{\hat{\theta}_{j',k'}}},$$

for $j, k \in \{1, \ldots, D_n\}$.

Next we discuss how problem (4.16) can be solved in practice. The first natural alternative is simply to use the ADMM algorithm from [13] as the objective function in (4.16) is the sum of convex functions (the surrogate

log-likelihood and the penalty). Thus, we introduce the slack variable z and rewrite the problem as

$$\begin{array}{rcll} \hat{\theta} & = & \underset{\theta}{\arg\min} & \sum_{j=1}^{D}\sum_{k=1}^{D}\left(e^{\theta_{j,k}} - x_{j,k}\,\theta_{j,k}\right) \;+\; \tau\|z\|_1, \\ & & \text{subeject to } z = \Delta^{(k+1)}\theta. & \end{array} \tag{4.17}$$

The ADMM proceeds with standard updates which we include in the appendix. The only difference with a typical ADMM implementation is that for the update of θ we consider a Taylor approximation around the previous iterate and so we have closed form update for θ at each iteration.

An alternative ADMM can be implemented in the same spirit of [110], by rewriting (4.16) as

$$\begin{array}{rcll} \hat{\theta} & = & \underset{\theta}{\arg\min} & \sum_{j=1}^{D}\sum_{k=1}^{D}\left(e^{\theta_{j,k}} - x_{j,k}\,\theta_{j,k}\right) \;+\; \tau\|\Delta^{(1)}z\|_1, \\ & & \text{subeject to } z = \Delta^{(k)}\theta. & \end{array} \tag{4.18}$$

and then exploiting the fused lasso trail decomposition algorithm from [136].

4.8 Examples and discussion

4.8.1 Comparison with kernel methods

We conducted a simulation study to examine the performance of histogram trend filtering versus some common methods for density estimation. Our first example is a three-component mixture of normals

$$f_1(y) = 0.9N(y \mid 0, 1) + 0.1N(y \mid -2, 0.1^2) + 0.1N(y \mid 3, 0.5^2)$$

shown in the top left panel of Figure 4.2. The second example is a five-component mixture of translated exponentials:

$$f_2(y) = \sum_{c=1}^{7} w_c \, Ex(y - m_c \mid 2) \,,$$

where the weight vector is $w = (1/7, 2/7, 1/7, 2/7, 1/7)$ and the translation vector is $m = (-1, 0, 1, 2, 3)$. Here $Ex(y \mid r)$ means the density of the exponential distribution with rate parameter r. This density is shown in the top right panel of Figure 4.2.

Our simulation study consisted of 25 Monte Carlo replicates for each of six different sample sizes: $n =$ 500, 1000, 2500, 5000, 10000, and 50000. For each simulated data set, we ran histogram trend filtering with $k = 1$ and $k = 2$. We benchmarked the approach against three other methods: kernel density estimation with the bandwidth chosen by five-fold cross-validation, kernel density estimation with the bandwidth chosen by the normal reference rule, and local polynomial density estimation with smoothing parameter chosen by cross-validation. In the reference-rule version of kernel density estimation, the bandwidth is chosen to be 0.9 times the minimum of the sample standard deviation and the interquartile range divided by $1.06n^{-1/5}$ [124]. We used the version of local polynomial density estimation implemented in the R package `locfit`.

Tables 4.1 and 4.2 show the average mean-squared error of reconstruction of all methods for both f_1 and f_2. Order-1 trend filtering has the lowest mean-squared error across all situations. Figure 4.2 provides a de-

Figure 4.2: Top two panels: the true densities f_1 (left) and f_2 (right) in the simulation study, together with samples of $n = 2500$ from each density. Middle two panels: results of histogram trend filtering for the f_1 sample (left) and the f_2 sample (right). Bottom two panels: results of kernel density estimation for the f_1 sample (left) and the f_2 sample (right). In the bottom four panels the reconstruction results are shown on a log scale.

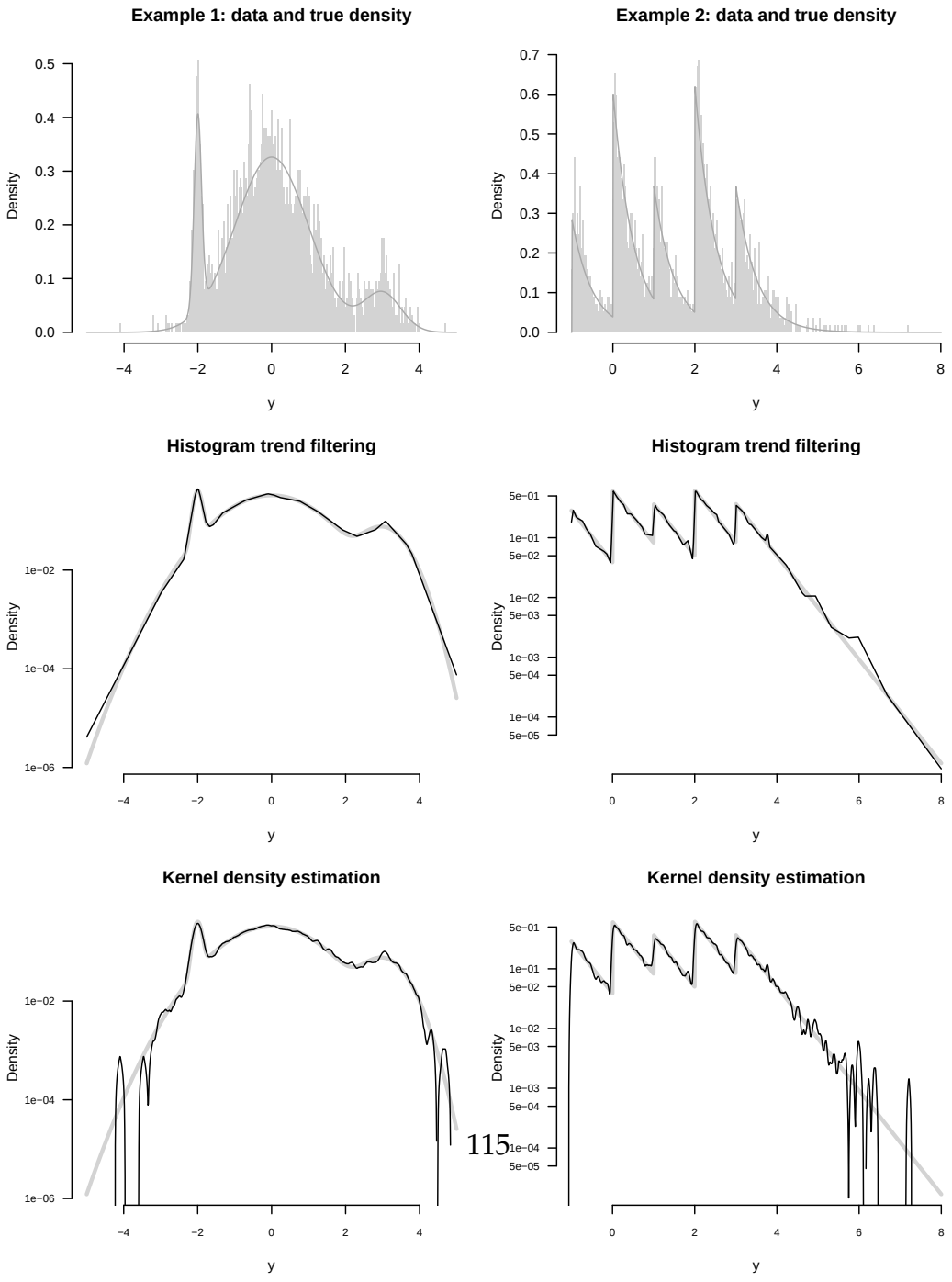


Table 4.1: Mean-squared error $\times$ 100 on example 1 for histogram trend filtering with $k = 1$ and $k = 2$ versus three other methods: kernel density estimation with bandwidth chosen by cross-validation, kernel density estimation using the normal reference rule, and local polynomial density estimation.

n	HTF ($k = 1$)	HTF ($k = 2$)	KDE (CV)	KDE (ref)	LP
500	2.5	4.9	3.1	4.0	3.3
1000	1.8	2.8	2.2	3.8	2.3
2500	1.3	1.6	1.7	3.3	1.6
5000	1.1	1.1	1.3	3.1	1.2
10000	0.7	0.7	0.9	2.8	0.9
50000	0.3	0.3	2.5	2.2	0.4

Table 4.2: Mean-squared error $\times$ 100 on example 2 for the same five methods in Table 4.1.

n	HTF ($k = 1$)	HTF ($k = 2$)	KDE (CV)	KDE (ref)	LP
500	5.7	6.8	5.5	8.8	6.2
1000	4.0	4.6	4.5	8.5	4.9
2500	3.0	3.3	3.7	7.9	3.5
5000	2.4	2.9	3.2	7.6	2.9
10000	2.0	2.9	2.8	7.0	2.6
50000	1.6	2.9	6.1	5.9	2.3

tailed look at the two simulated data sets. The top two panels show f_1 and f_2 together with a single simulated data set of $n = 2500$ from each density. The middle two panels show the reconstruction results for histogram trend filtering with $k = 1$, while the bottom two panels show the reconstruction results for kernel density estimation with the bandwidth chosen by cross validation. The trend-filtering estimator shows excellent adaptivity; it captures the sharp jumps in each of the true densities, without suffering from pronounced under-smoothing in other regions.

Table 4.3: Mean-squared error $\times$ 10 on example 1, averaging over 50 MC simulations

n	HTF ($k = 1$)	RU	TV	Taut string	W-N	MAPT	Bayesian HTF ($k = 1$)
500	1.2	11.1	3.4	3.4	3.7	1.3	**1.0**
1000	0.9	9.6	2.4	3.8	1.1	0.9	**0.8**
2000	**0.4**	7.3	1.2	1.7	0.5	0.5	0.6
3000	**0.3**	2.8	0.9	1.1	0.4	0.4	**0.3**
4000	**0.2**	1.6	0.7	0.7	**0.2**	0.3	0.3

4.8.2 Comparison with other adaptive and penalized methods

So far we have considered comparisons of HTF versus estimation methods that scale well with the number of samples. We now conclude with examples comparing our HTF against other penalized methods that face problems with large numbers of samples, or approaches that enjoy adaptivity to different degrees of smoothness of the true density. These methods are the the penalized likelihood approach from [150] (W-N), the total variation approach from [120] (TV) using their universal penalty, the taut string method from [34] (which is closely related to the estimator from [120]), the root–unroot algorithm for density estimation via wavelet block thresholding from [17] (RU), and the Markov adaptive Pólya tree method from [96] (MAPT).

Using the methods described above as benchmarks, we assess the performance of our histogram trend filtering selecting the parameter as in Section 4.5, and also using the Bayesian perspective from Section 4.6. To

Table 4.4: Mean-squared error $\times$ 100 on example 2, averaging over 50 MC simulations.

n	HTF $(k=1)$	RU	TV	Taut string	W-N	MAPT	Bayesian HTF $(k=1)$
500	3.4	4.1	6.0	10.0	1.3	5.6	**1.0**
1000	1.6	2.3	3.6	5.4	**0.8**	3.5	**0.8**
2000	1.1	1.7	2.2	3.2	**0.3**	2.2	0.6
3000	0.9	1.4	1.8	2.5	**0.2**	1.6	0.3
4000	**0.2**	1.3	1.5	2.3	**0.2**	1.2	0.3

Table 4.5: Mean-squared error $\times$ 10 on example 3, averaging over 50 MC simulations.

n	HTF $(k=1)$	RU	TV	Taut string	W-N	MAPT	Bayesian HTF $(k=1)$
500	**1.7**	23.8	6.1	5.3	1.8	2.4	1.9
1000	**1.1**	17.8	3.4	2.1	0.9	1.4	1.6
2000	**0.5**	5.6	2.1	0.9	0.6	0.9	0.7
3000	**0.4**	4.5	1.4	0.7	0.5	0.6	0.5
4000	**0.3**	1.7	1.0	0.5	0.4	0.4	0.4

evaluate the quality of these methods, we borrow some challenging density estimation examples from the literature. Particularly, Example 1 in Figure 4.3 is taken from [150], and Examples 2, 3 and 4 in Figure 4.3 are taken from [96].

For all the methods that require binning the data we choose $D_n = 2^{10}$ to be fixed, and compute the mean square error of the estimates on the centers of evenly spaced bins of size D_n^{-1}. This is the measure of performance that we use in order to have a fair comparison between the different meth-

Table 4.6: Mean-squared error $\times$ 10 on example 4, averaging over 50 MC simulations.

n	HTF ($k = 1$)	RU	TV	Taut string	W-N	MAPT	Bayesian HTF ($k = 1$)
500	**1.8**	26.0	8.7	5.3	3.5	2.7	2.0
1000	**1.1**	22.0	5.2	2.1	1.5	1.7	**1.1**
2000	**0.6**	6.3	2.9	0.9	0.9	1.0	**0.6**
3000	**0.4**	2.2	2.0	0.7	0.7	0.7	0.5
4000	**0.3**	1.9	1.5	0.5	0.7	0.6	0.4

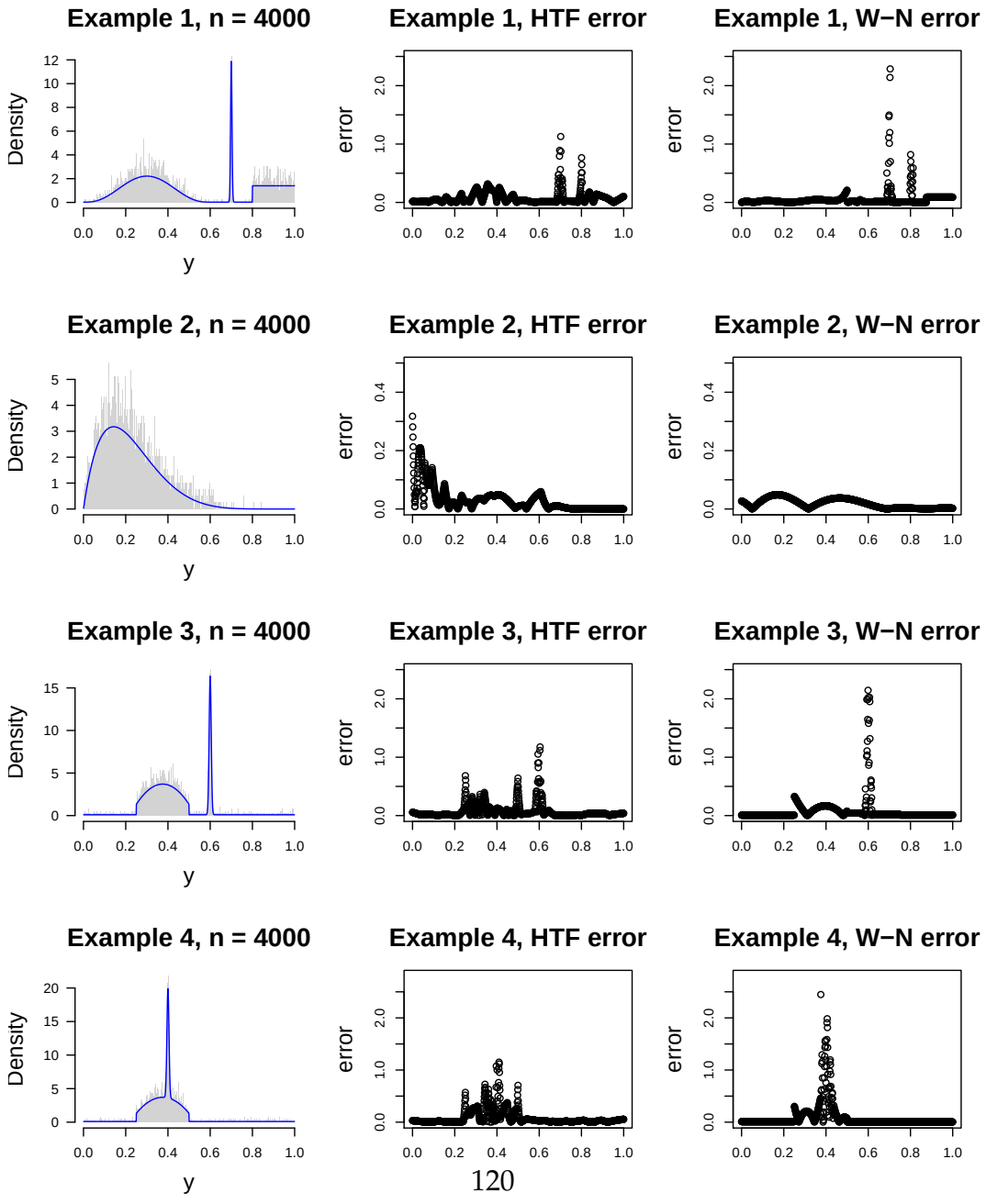
Table 4.7: Time in seconds for Example 1, averaging over 50 MC simulations.

n	HTF ($k = 1$)	RU	TV	Taut string	W-N
500	12.2	0.04	7.4	0.01	1.1
1000	10.4	0.03	20.2	0.02	4.4
2000	8.3	0.03	45.1	0.05	22.1
3000	6.9	0.03	85.2	0.05	37.2
4000	6.3	0.03	135.6	0.07	117.4

ods.

Considering different sample sizes and the densities in Figure 4.3, the results in Tables 4.3, 4.4, 4.5, and 4.6 show that, generally, histogram trend filtering outperforms the competing approaches. This is particularly true in the examples involving true densities which present different degrees of smoothness in domain. However, in the case of Example 2 which is just a beta distribution and not a mixture, we observe that the method W-N is in most cases better than our HTF, but even then HTF is still quite competitive.

Figure 4.3: Each row of panels above represents an example considered in this section. For each row the first column shows the true density along with $n = 4000$ draws from the respective density. The second column shows the error for the solution given by HTF(k=1). Similarly, the third column of panels represents the estimate error for W-N.



To give a visual comparison between W-N and HTF, Figure 4.3 illustrates that for Examples 1,3, and 4, HTF captures the peaks of the true density better than W-N. This fact then reflects in better performance by HTF.

On the other hand, we also emphasize that despite its accuracy, HTF also enjoys low computational complexity. This is seen in Table 4.7 where we notice that HTF is much more computationally efficient than W-N, which as discussed above was the most competitive method in our experiments. An additional interesting feature in Table 4.7 is that HTF is faster as n increases and the number of bins remains fixed. Suggesting the optimization problem (4.3) becomes easier as D_n remains fixed and n increases.

4.8.3 NYC Taxi Data

Our final experiment consists of a prediction task of 2D density estimation. We use data from the NYC Taxi & Limousine Commission, which consists of information about trips of taxis in New York city.

In order to explore an interesting 2D density estimation problem, we construct a matrix $X \in \mathbb{R}^{5000\times 2}$ where each column represents one of two variables: fare amount of a taxi trip and the respective distance of the trip (these have both been normalized to be between 0 and 1). Moreover, the rows of X represent different taxi trips along different locations of New York city. We selected the trips that happened in June 1, 2016 between roughly 2:45 am and 6:00 pm, a normal day in New York, NY.

Figure 4.4: The left panel shows the binned counts for a training set consisting of 75% of the subset of the NYC Taxi data that we used. The right panel shows the counts for the respective test set or remaining 25% of the data.

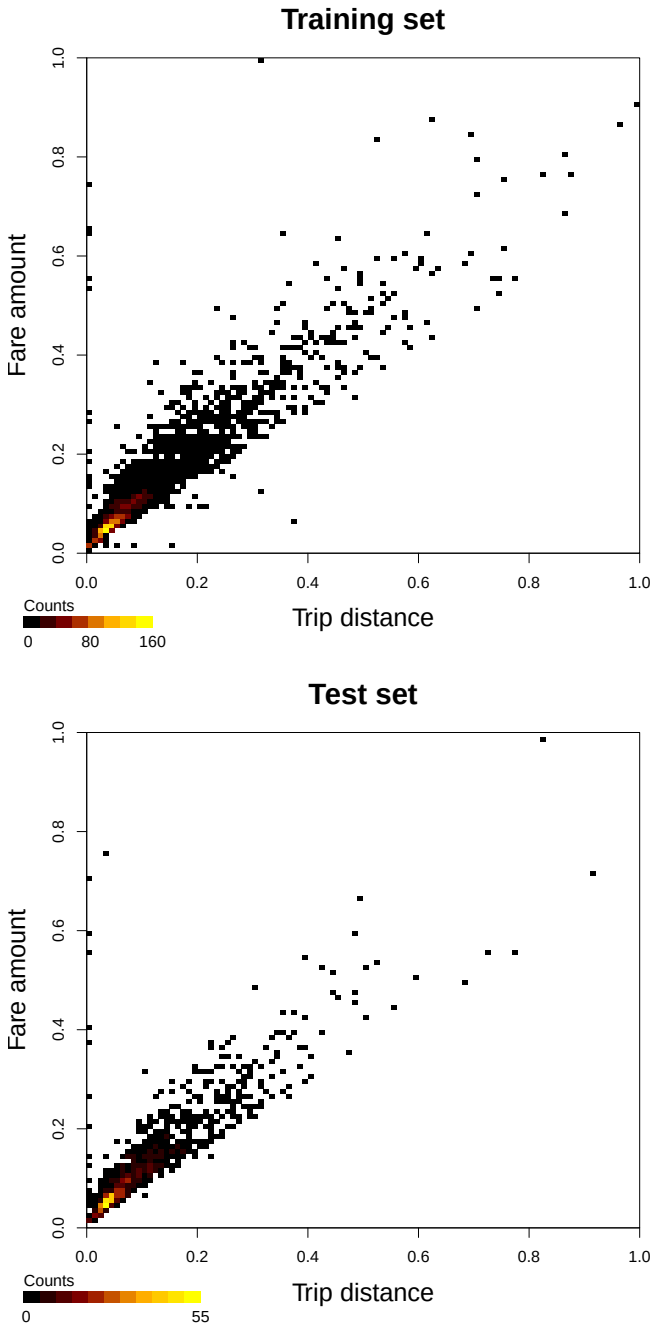


Table 4.8: Average log-likelihood on test set times 10^{-3}, averaging over 50 random training and test sets of sizes $s\%$ and $(100-s)\%$ respectively.

(100-s)%	HTF ($k=0$)	HTF($k=1$)	Multivariate Kernel
15%	2.58	2.73	2.33
20%	3.11	3.61	3.11
25%	4.14	4.52	3.95
30%	5.02	5.42	4.39

To evaluate the performance of our HTF on the data set described a above, we consider splitting the data X into training and test set. Thus, for different choices of s, we randomly choose $s\%$ of the rows of X for training, and $(100-s)\%$ for testing. See Figure 4.4 for an example of training and testing sets. There, for each of these two sets, we display the counts after binning the data according to the Section 4.7 with $D_n = 100$.

We then measure the performance of different methods, fitted on the training set, by simply computing the log-likelihood of the corresponding test set. The methods we consider are our HTF with $k = 0, 1$, and multivariate kernel density estimation. For the latter, we select the bandwidth matrix as in [49], while for our HTF methods we select the tuning parameter by BIC. The proposed BIC is based on using Lemma 1 from [149], where the authors provided an unbiased estimator of the degrees of freedom for trend filtering problems. However, this does not necessarily provide and unbiased estimator in our context, although it seems to give good results based on our experience.

After considering splitting X into multiple training and test sets, we

obtain the results illustrated in Table 4.8. These show that our HTF method outperforms the standard and widely used multivariate kernel density estimation, hence, once again showing the value of our approach.

4.9 Conclusion

In summary, we have shown that histogram trend filtering can be successfully applied to the problem of density estimation. This estimator enjoys both computational and theoretical attractive properties. On the computational side, our experiments suggests that histogram trend filtering scales remarkably well with sample size, and that in practice it is just as computationally efficient as widely used methods based on kernel density estimation (KDE). However, unlike such methods, histogram trend filtering does not suffer from simultaneous over- and under-smoothing. Rather, our estimator can easily adapt to different levels of smoothness of the unknown true density.

Many methods have been proposed in the literature to deal with the problem of local adaptivity, e.g [150, 120]. As we have shown, these methods face challenges specifically in regions where the smoothness of true density changes rapidly. We have shown that histogram trend filtering can better adjust to such situations, while overcoming the scalability problems also inherent in other penalized methods. Thus histogram trend filtering enjoys both the computational efficiency of KDE methods and the adaptive properties of penalized estimators. Finally, our risk bound provides

strong theoretical guarantee of good performance for histogram trend filtering. This combination of practicality with strong statistical guarantees makes histogram trend filtering an ideal candidate for use in routine data-analysis applications that call for a quick, efficient, accurate density estimate.

Chapter 5

A deconvolution path for mixtures

The contents of this chapter are based on the working paper [97].

5.1 Deconvolution in mixture models

Suppose that we observe $y = (y_1, \ldots, y_n)$ from the model

$$y_i \mid \mu_i \sim \phi(y_i \mid \mu_i)\,, \quad \mu_i \overset{i.i.d.}{\sim} f_0\,, \tag{5.1}$$

where $\phi(\cdot \mid \mu)$ is a known distribution with location parameter μ, and f_0 is an unknown mixing distribution. Marginally, we have specified a mixture model for y_i:

$$m(y_i) = \int_{\mathbb{R}} \phi(y_i - \mu_i)\; f_0(\mu_i)\; d\mu_i = (\phi * f_0)(y_i)\,. \tag{5.2}$$

The problem of estimating the mixing distribution f_0 is commonly referred to as *deconvolution*: we observe draws from the convolution $m = \phi * f_0$, rather than from f_0 directly, and we wish to invert this blur operation to recover the distribution of the latent location parameters. Models of this form have been used in a wide variety of applications and have attracted significant attention in the literature [e.g. 78, 58, 56, 107, 63]. Yet the estimation of

f_0 continues to pose both theoretical and practical challenges, making it an active area of statistical research [e.g. 39, 52, 40].

In this chapter, we propose a nonparametric method for deconvolution that is both statistically and computationally efficient. Our method can be motivated in terms of an underlying Bayesian model incorporating a prior into model (5.1), but it does not involve full Bayes analysis. Rather, we use a two-step "bin and smooth" procedure. In the "bin" step, we form a histogram of the sample, yielding the number of observations x_j that fall into the jth histogram bin. In the "smooth" step, we use the counts x_j to compute a maximum *a posteriori* (MAP) estimate of f_0 under a prior that encourages smoothness.

We show that this nonparametric empirical-Bayes procedure yields excellent performance for deconvolution, at reduced computational cost compared to full nonparametric Bayesian methods. Our main theorems establish conditions under which the method yields a consistent estimate of the mixing distribution f_0, and provide a concentration bound for recovery of the marginal distribution m. We also provide simulation evidence that the method offers practical improvements over existing state-of-the-art methods.

5.1.1 Methodological issues in deconvolution

To complement these theoretical results, we address two main methodological themes. First, we emphasize the importance of sensitivity analysis:

that is, characterizing how the deconvolution estimate changes with respect to the assumed smoothness of f_0. Inverting the blur operation $m = \phi * f_0$ is typically ill-posed, in that large changes in f_0 produce only small changes in m. The role of prior assumptions matters a great deal here. But for many methods, the mapping between tuning parameters and the smoothness of the estimate is not apparent. Our solution-path approach makes this mapping very explicit, since it returns a range of estimates that are all compatible with a given marginal.

The secondary theme we emphasize is the connection between deconvolution (5.1) and the canonical normal-means problem. Here $(y_i \mid \mu_i) \sim N(\mu_i, 1)$, and the object of inferential interest is the vector of means $(\mu_1, \ldots, \mu_n)$ rather than the mixing measure f_0. A classic result known as Tweedie's formula [114, 51] makes this connection explicit in exponential-family models. Although our main goal is to provide a good estimate for f_0, we also highlight the advantages of using our method in conjunction with Tweedie's formula in the normal-means problem. This is made explicit in the experiments sections.

5.2 Connections with previous work

We first recall the work by [78], who consider estimating f_0 using the nonparametric maximum likelihood estimation. The Kiefer–Wolfowitz estimator (KW) has some appealing features: it is completely nonparametric and invariant to translations of the data, it requires no tuning parameters,

and it is consistent under fairly general conditions. Balanced against these desirable features is one significant disadvantage: $\hat{f}$ is a discrete distribution involving as many as $n+1$ point masses. [83] refer to this phenomenon as a "Dirac catastrophe," for the reason that in most settings f_0 will be relatively smooth, and the discreteness of the KW estimator is unappealing. A related consistent estimator was studied in [62].

Deconvolution has also been studied using Bayesian methods. In the context of repeated measurements, or multivariate deconvolution, we highlight recent work by [121, 122, 131]. Moreover, for the one dimensional density estimation problem, a flexible choice is the Dirichlet Process (DP) studied in [58] and [55]. For a Dirichlet prior in deconvolution problems, concentration rates were recently studied in [43]. Related models were considered by [42] and [105] for finite mixture of normals.

The DP provides a very general framework for estimating the mixing density f_0. However, as [103] argue, fitting a Dirichlet process mixture does not scale well with the number of observations n. For microarray studies, n ranges from thousands to tens of thousands, whereas for more recent studies of fMRI data or single-nucleotide polymorphisms, n can reach several hundreds of thousands [e.g. 135]. For such massive data sets, fitting a DP mixture model can be very time-consuming.

To overcome this difficulty, [107], [143], [104], and [103] studied a predictive recursive (PR) algorithm. The resulting estimator scales well with large data sets while remaining reasonably accurate, thereby solving one

the main challenges faced by the fully Bayesian approach.

Finally, we note the work by [21, 132, 153, 56, 57, 66, 20, 39], among others, who considered kernel estimators. Their idea is motivated by (5.1) after taking the Fourier transform of the corresponding convolution of densities, then solving for the unknown mixing density using kernel approximations for the Fourier transform of the true marginal density. The resulting kernel estimator enjoys attractive theoretical properties: for each $\mu_0 \in \mathbb{R}$, the estimator has optimal rates of convergence towards $f_0(\mu_0)$ for squared-error loss when the function f_0 belongs to a smooth class of functions [56].

5.3 A deconvolution path

5.3.1 Overview of approach

We now described our proposed approach in detail. We study deconvolution estimators related to the variational problem

$$\underset{f}{\text{minimize}} \quad -\sum_{i=1}^{n} \log(\phi * f)(y_i) \quad \text{subject to} \quad \int_{\mathbb{R}} f(\mu)\, d\mu = 1, \quad J(f) \leq t, \tag{5.3}$$

where $J(f)$ is a known penalty functional. The choices of J we consider include ℓ_1 or ℓ_2 penalties on the derivatives in the log-space to encourage smoothness:

$$\| \log f^{(k)} \|_s^q \;\; = \;\; \int_{\mathbb{R}} | \log f^{(k)}(\mu) |^s \, d\mu, \tag{5.4}$$

with $s = q = 1$ or $s = q = 2$ and where $\log f^{(k)}$ is the derivative of order k of the log prior. The penalty involving the first derivative is an especially

interpretable one, as $d \log f(\mu)/d\mu = f'(\mu)/f(\mu)$ is the score function of the mixing density.

Note that an alternative interpretation of our approach is as a MAP estimator. To see this we consider the (possibly improper) prior on the mixing density

$$p(f) \propto \exp\left(-J(f)\right) \mathbb{I}\left(f \in \mathcal{A}\right),$$

where $\mathcal{A}$ is an appropriate class of density functions. The posterior distribution is $p(f \mid y) \propto p(y \mid f)p(f)$, and our MAP estimator therefore solves, for an appropriate $\tau > 0$,

$$\text{argmin}_{f \in \mathcal{A}} - \log p(y \mid f) + \frac{\tau}{2} J(f). \tag{5.5}$$

This belongs to a general class of MAP estimators that have been studied in [65] and [127] for the classical problem of density estimation. For deconvolution problems we note the recent work by [147] which penalizes the marginal density rather than the derivatives of the mixing density as we propose. Moreover such penalization is not motivated to encourage smoothness. Rather, it is an ℓ_2 projection on to the space of acceptable marginal densities. An alternative penalized likelihood method was studied in [91] in the different context where the marginal density has atoms. There the authors use the roughness penalty $J(f) = \int |f'(\mu)|^2 d\mu$ which differs from our approach that penalizes the log-mixing density and hence ensures that the solutions will be positive. Also, we allow different degrees of smoothness depending on the choice of k. Moreover, while [91] only con-

sidered sample sizes in the order of hundreds, we show in the next sections that our estimator can scale to much larger data sets while still enjoying attractive statistical properties.

More recently, [52] proposed a penalized approach to deconvolution that is also based on regularization but differs from ours. Such approach proceeds by assuming that the mixing density is discrete with support $\{\theta_1, \ldots, \theta_N\}$. Then [52] specified a parametric model on the mixing density of the form

$$f(\theta_j) \;=\; \exp\left(Q_{j,\cdot}^T \alpha - c(\alpha)\right), \quad j \;=\; 1, \ldots, N,$$

where $Q_{j,\cdot}$ is the $j-th$ row of the matrix $Q \in \mathbb{R}^{N \times p}$, for some $p > 0$, which is used to encourage structure on the mixing density. Moreover, $c(\alpha)$ is a normalizing constant satisfying

$$c(\alpha) \;=\; \log\left(\sum_{j=1}^{N} \exp\left(Q_{j,\cdot}^T\, \alpha\right)\right).$$

Then summing over all the $\{\theta_j\}_{j=1}^N$ and considering the contribution of the different samples, [52] arrives to the minus log-likelihood

$$l(\alpha) \;=\; -\sum_{i=1}^{n} \log\left(\sum_{j=1}^{N} \mathrm{N}(y_i|\theta_j)\, \exp\left(Q_{j,\cdot}^T \alpha - c(\alpha)\right)\right).$$

The estimator from [52] results from solving

$$\underset{\alpha}{\text{minimize}}\; l(\alpha) \;+\; c_0 \left(\sum_{h=1}^{p} \alpha_h^2\right)^{1/2}. \tag{5.6}$$

where c_0 is either 1 or 2.

We note that the estimator (5.6) is possibly limited by the following aspects. First, the choice of the matrix Q, which [52] recommends to be a spline basis representation, will produce estimates that suffer from local adaptivity problems. Moreover, there is no theoretical support of choosing $c_0 \in \{1, 2\}$. In our experiments section we will present experimental comparisons between the estimator (5.6) and our approach.

Finally, we emphasize that our approach is not, in any sense, related to the ridge parameter deconvolution estimator from [66]. Such estimator does not penalizes that log-likelihood as we propose, but rather is designed to avoid the need to choose a kernel function, in the original kernel estimator from [56], by ridging the integral in its definition with a positive function.

5.3.2 Binned counts problem

Throughout this section we assume that ϕ corresponds to the pdf of the standard normal distribution, although the arguments can easily be generalized to other distributions.

To make estimation efficient in scenarios with thousands or even millions of observations, we actually fit a MAP estimator based on binning the data. First, we use the sample to form a histogram $\{I_j, x_j\}_{j=1}^{D}$ with D bins, where I_j is the $j-$th interval in the histogram and $x_j = \#\{y_i \in I_j\}$ is the associated count. For ease of exposition, we assume that the intervals take the form $I_j = \xi_j \pm \Delta/2$, i.e. have midpoints ξ_j and width Δ, although this

is not essential to our analysis. To arrive to a discrete estimator, instead of Problem (5.3), we consider an approximation, a reparametrization $g = \log f$, and put the penalty in the objective function with a regulariation parameter $\tau > 0$,

$$\underset{g \in R^D}{\text{minimize}} \quad -\tfrac{1}{n} \textstyle\sum_{j=1}^{D} x_j \log\left(\, \phi * e^g(\xi_j)\right) + \tfrac{\tau}{2}\, J(e^g) \quad \text{subject to} \quad \textstyle\int e^{g(\mu)} d\mu = 1. \tag{5.7}$$

We then approximate (5.7) by solving

$$\begin{array}{ll} \underset{g \in R^D}{\text{minimize}} & -\tfrac{1}{n} \textstyle\sum_{j=1}^{D} x_j \log\left(\sum_{i=1}^{D} \Delta\, \phi(\xi_j - \xi_i) e^{g_i} \right) + \tfrac{\tau}{2}\, \|\Delta^{(k+1)} g\|_q^s \\ \text{subject to} & \textstyle\sum_{i=1}^{D_n} \Delta\, e^{g_i} = 1, \end{array} \tag{5.8}$$

where $s = q = 1$ or $s = q = 2$, and $\Delta^{(k+1)}$ is the k-th order discrete difference operator. Concretely, when $k = 0$, $\Delta^{(1)}$ is the $(D-1) \times D$ matrix encoding the first differences of adjacent values:

$$\Delta^{(1)} = \begin{pmatrix} 1 & -1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & -1 & 0 & \cdots & 0 \\ \vdots & & & & \ddots & \vdots \\ 0 & \cdots & & 0 & 1 & -1 \end{pmatrix}. \tag{5.9}$$

For $k \geq 1$, $\Delta^{(k+1)}$ is defined recursively as $\Delta^{(k+1)} = \Delta^{(1)}\Delta^{(k)}$, where $\Delta^{(1)}$ from (5.9) is of the appropriate dimension. Thus when $k = 0$, we penalize the total variation of the vector θ [c.f. 117, 137] and should expect estimates that are shrunk towards piecewise-constant functions. When $k \geq 1$, the estimator penalizes higher-order versions of total variation, similar to the polynomial trend-filtering estimators studied by [140].

Interestingly, following the proof of Theorem 4.3.1 we find that (5.8)

is equivalent to

$$\underset{\theta \in \mathcal{R}^D}{\text{minimize}} \quad l(\theta) + \frac{\tau}{2} \|\Delta^{(k+1)}\theta\|_q^s \,, \tag{5.10}$$

where

$$l(\theta) = \sum_{j=1}^{D} \left\{ \lambda_j(\theta) - x_j \log \lambda_j(\theta) \right\} \,,$$

with $\lambda_j = \sum_{j=1}^{D} G_{ij} e^{\theta_i}$, $G_{ij} = \Delta\, \phi(\xi_j - \xi_i)$, and $\hat{\theta}$ solves (5.10) if only if $\hat{\theta} - \log(n\,\Delta)\mathbf{1}$ solves (5.8). Hence, in practice we solve the unconstrained optimization Problem (5.10).

5.3.3 Solution algorithms

We start discussing the implementation details for solving (5.10) in the case $s = q = 1$. To solve this problem, motivated by the work on trend filtering for regression by [110], we rewrite the problem as

$$\underset{\theta}{\text{minimize}} \quad l(\theta) + \tfrac{\tau}{2} \|\Delta^{(1)}\alpha\|_1 \quad \text{subject to} \quad \alpha = \Delta^{(k)}\theta. \tag{5.11}$$

Next we proceed via the alternating-direction method of multipliers (ADMM), as in [110]. [See 13, for an overview of ADMM.] By exploiting standard results we arrive at the scaled augmented Lagrangian corresponding to the constrained problem (5.11):

$$L_\rho(\theta, \alpha, u) = l(\theta) + \frac{\tau}{2} \|\Delta^{(1)}\alpha\|_1 + \rho u^T \left(\alpha - \Delta^{(k)}\theta\right) + \frac{\rho}{2} \|\alpha + u - \Delta^{(k)}\theta\|_2^2 \,.$$

This leads to the following ADMM updates at each iteration j:

$$\begin{aligned} \theta^{j+1} &\leftarrow \underset{\theta}{\text{argmin}} \left(l(\theta) + \tfrac{\rho}{2} \left\| \alpha^j + u^j - \Delta^{(k)}\theta \right\|_2^2 \right) , \\ \alpha^{j+1} &\leftarrow \underset{\alpha}{\text{argmin}} \left(\tfrac{1}{2} \|\alpha - \Delta^{(k)}\theta^{j+1} + u^j\|_2^2 + \tfrac{\tau}{2\rho} \|\Delta^{(1)}\alpha\|_1 \right) \\ u^{j+1} &\leftarrow u^j + \alpha^{j+1} - \Delta^{(k)}\theta^{j+1}. \end{aligned} \tag{5.12}$$

Note that in (5.12) the update for θ involves solving a sub-problem whose solution is not analytically available. To deal with this, we use the well known Broyden–Fletcher–Goldfarb–Shanno (BFGS) algorithm, which is very efficient because the gradient of the θ sub-problem objective is available in closed form. The update for α can be computed in linear time by appealing to the dynamic programming algorithm from [74].

In the case $p = q = 2$, both components of the objective function in (5.10) have closed-form gradients, see the appendix. Thus we can solve the problem using any algorithm that can use function and gradient calls. In particular, we use BFGS.

5.3.4 Solution path and model selection

One of the major advantages of our approach is that it yields an entire deconvolution path, comprising a family of estimates $\hat{f}(\tau)$ over a grid of smoothness parameters. This path is generated efficiently using warm starts. We initially solve (5.11) for a large value of τ, for which the result estimate is nearly constant. We then use this solution to initialize the ADMM at a slightly smaller value of τ, which dramatically reduces the computation time compared to an arbitrarily chosen starting point. We proceed iteratively until solutions have been found across a decreasing grid of τ values (which are typically spaced uniformly in $\log \tau$).

The resulting deconvolution path can be used to inspect a range of plausible estimates for f_0, with varying degrees of smoothness. This allows

the data analyst to bring to bear any further prior information (such as the expected number of modes in f_0) that was not formally incorporated into the loss function. It also enables sensitivity analysis with respect to different plausible assumptions about the smoothness of the mixing distribution. We illustrate this approach with a real-data example in Section 5.4.

However, in certain cases—for example, in our simulation studies—it is necessary to select a particular value of τ using a default rule. We now briefly describe heuristics for doing so based on ℓ_1 and ℓ_2 penalties with $k = 1$. These heuristics are used in our simulation studies. For the case of ℓ_1 regularization, motivated by [142], we consider a surrogate AIC approach by computing

$$\text{AIC}_\tau = l(\hat{\theta}_\tau) + k + 1 + \left| \left\{ i : (\Delta^{(k+1)} \hat{\theta}_\tau)_i \neq 0 \right\} \right|,$$

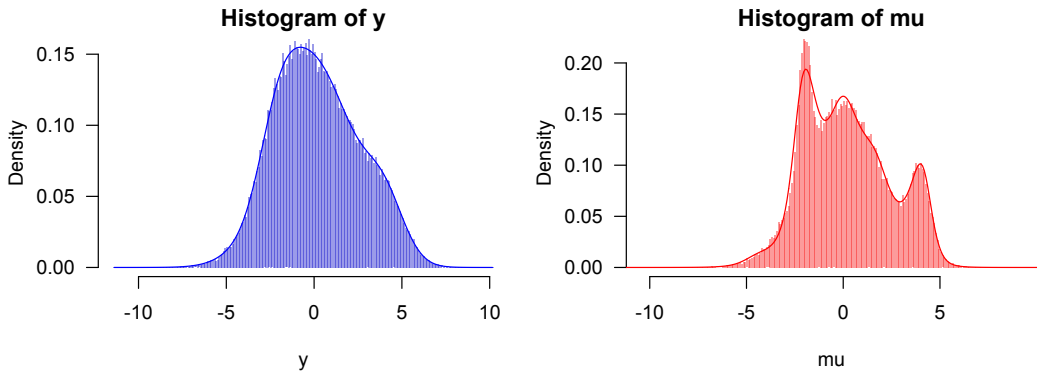
and choosing the value of τ that minimizes this expression. Here, $\hat{\theta}_\tau$ denotes the solution given by L1-D with regularization parameter τ.

In the case of ℓ_2 regularization the situation is more difficult, since there is not an intuitive notion of the number of parameters of the model. Instead, we consider an ad-hoc procedure based on cross validation. This solves the problem for a grid of regularization parameters and chooses the parameter the minimizes $l(\hat{\theta}_\tau^{\text{held out}}) + \|\Delta^{(k+1)} \hat{\theta}_\tau^{\text{held out}}\|_1$, where $\hat{\theta}_\tau^{\text{held out}}$ is defined as

$$\hat{\theta}_\tau^{\text{held out}} = \hat{\theta}_\tau - \log(n\Delta) + \log(n_{\text{held out}}\Delta),$$

with $\hat{\theta}_\tau$ the solution obtained by fitting the model on the training set which

Figure 5.1: Example of deconvolution with an ℓ^2 penalty on the discrete first derivative ($k = 1$). The left panel shows the data histogram together with the fitted marginal density as a solid curve. The right panel shows the histogram of the μ_i's together with the estimated mixing measure as a solid curve.



consists of 75% of the data. Here, $l(\hat{\theta}_\tau^{\text{held out}})$ is evaluated using the counts from the held out set which has 25% of the data, and $n_{\text{held out}}$ is the number of observations in such set. Our motivation for using the additional term $\|\Delta^{(k+1)}\hat{\theta}_\tau^{\text{held out}}\|_1$ is that ℓ_0 works well when the problem is formulated with ℓ_1 regularization. However, when (5.10) is formulated using ℓ_2, the penalty ℓ_0 is not suitable so instead we use ℓ_1. Our simulations in the experiments section will show that this rule works well in practice.

5.3.5 A toy example

We conclude this section by illustrating the accuracy of our regularized deconvolution approach on a toy example. In this example we draw 10^5 samples $\{y_i\}$ with the corresponding $\{\mu_i\}$ drawn from a mixture of three normal distributions. Figure 5.1 shows the samples of both the observations

y_i (left panel) and the means μ_i (right panel), together with the reconstructions provided by our method. Here, we solve the ℓ^2 version of problem (5.10) by using the BFGS algorithm and choose τ using the heuristic just described.

It is clear that regularizing with an ℓ_2 penalty provides an excellent fit of the marginal density. Surprisingly, it can also capture all three modes of the true mixing density, a feature which is completely obscured in the marginal). Our experiments in Section 5.6.1 will show in a more comprehensive way that our method far outperforms other approaches in its ability to provide accurate estimates for multi-modal mixing distributions.

5.4 Sensitivity analysis across the path

In this section, we provide an example of a sensitivity analysis using our deconvolution path estimator. We examine data originally collected and analyzed by [129] on gene expression for 12,600 genes across two samples: 9 healthy patients and 25 patients with prostate tumors. The data come as a set of 12,600 t-statistics computed from gene-by-gene tests for whether the mean gene-expression score differs between the two groups. After turning these 12,600 t-statistics into z-scores via a CDF transform, we estimate a deconvolution path assuming a Gaussian convolution kernel. We use an ℓ^2 penalty and a grid of τ values evenly spaced on the logarithmic scale between 10^7 and 10^{-3}.

Figure 5.2: Rows A–E show five points along the deconvolution path for the prostate cancer gene-expression data. The regularization parameter is largest in Row A and gets smaller in each succeeding row.

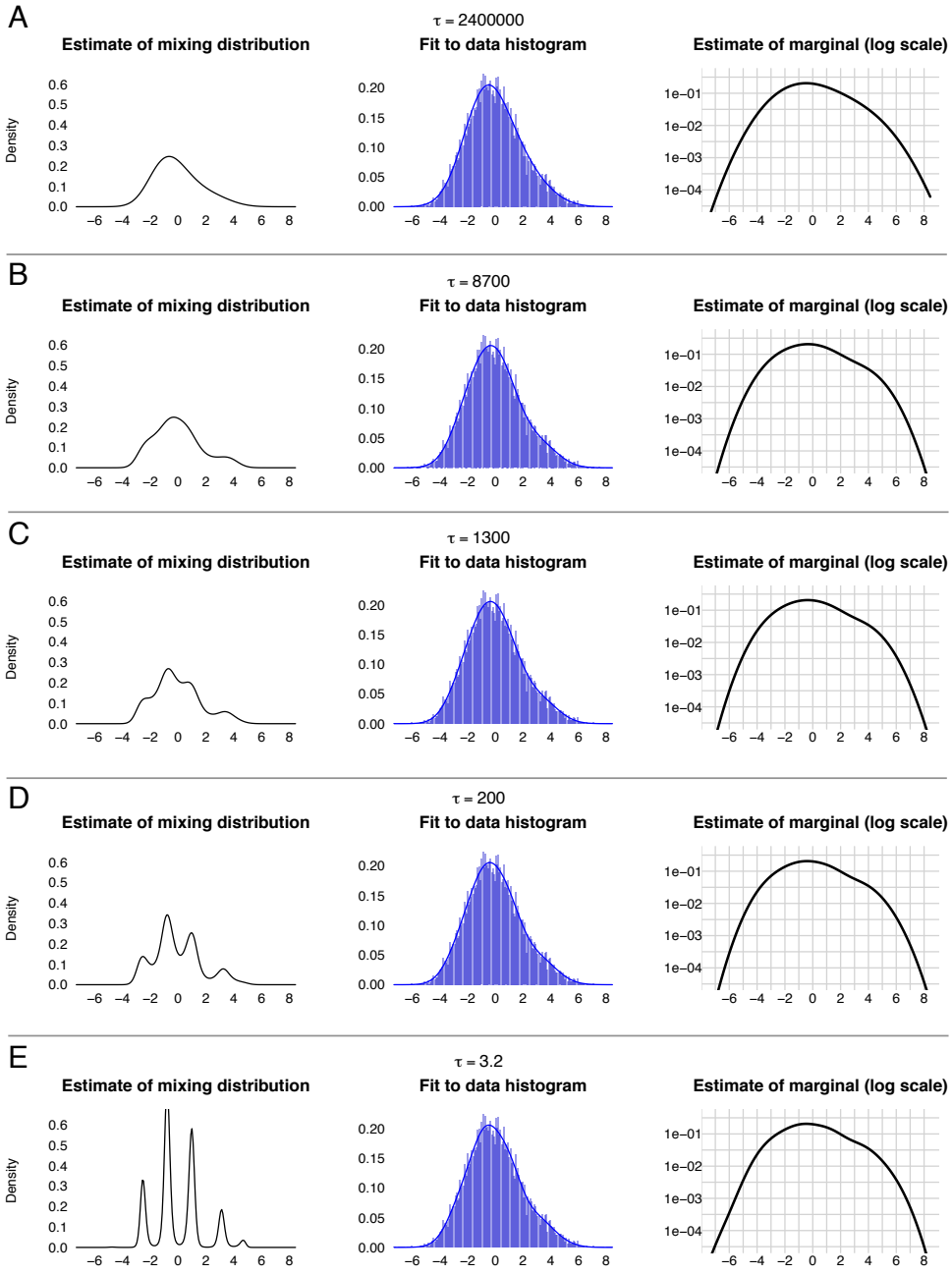
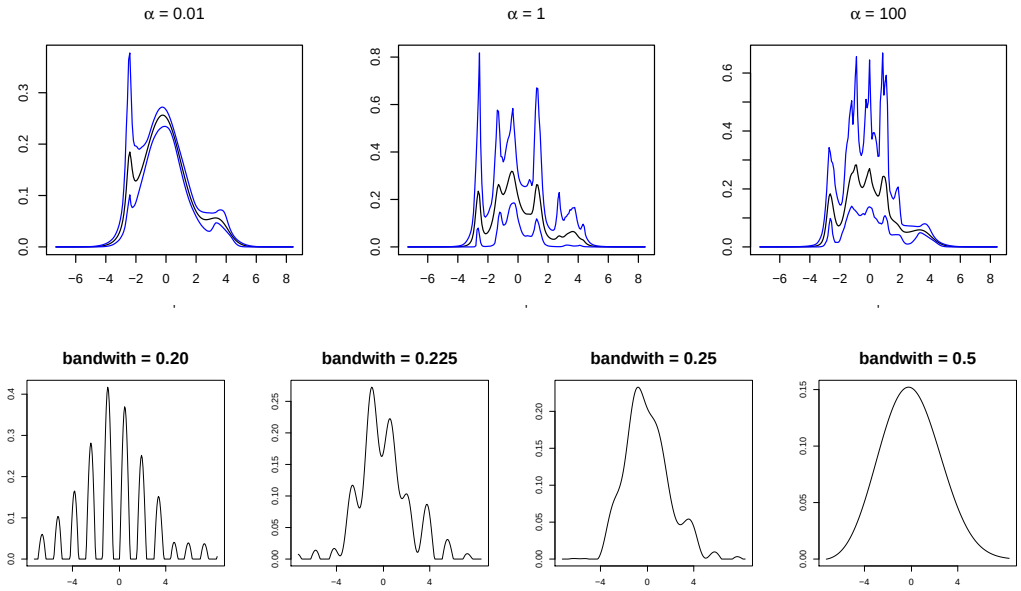


Figure 5.3: The first three panels show 95% confidence bands and posterior mean from 15000 posterior samples from a mixture of 10 normals prior on the latent variables μ. Panels 4-7 then shows the estimated mixing density using the kernel estimator with different bandwidth choices.



Each row of Figure 5.2 shows five points along the deconvolution path; the regularization parameter is largest in Row A and gets smaller in each successive row. Within each row, the left column shows the estimated mixing distribution $\hat{f}$ for the given value of τ. The middle column shows the histogram of the data together with the fitted marginal density $\hat{m} = \phi * \hat{f}$. The right column shows the fitted marginal density on the log scale, with a regular grid to facilitate comparison of the results across different values of τ.

The figure shows that, while the estimate of the mixing distribu-

tion changes dramatically across the deconvolution path, the estimate of the marginal density is much more stable. Even on the log scale (right column), the differences among the fitted marginal densities are not visually apparent in Panels B through E, even as the regularization parameter varies across three orders of magnitude.

This vividly demonstrates the well-known fact that deconvolution, especially of a Gaussian kernel, is a very ill-posed inverse problem. There is little information in the data to distinguish a smooth mixing distribution from a highly multimodal one, and the model-selection heuristics described earlier are imperfect. A decision to prefer Panel B to Panel E, for instance, is almost entirely due to the effect of the prior. Yet for most common deconvolution methods, the mapping between prior assumptions and the smoothness of the estimate is far from intuitive. By providing a full deconvolution path, our method makes this mapping visually explicit.

For reference, it is interesting to compare our deconvolution path to the results of other methods. Figure 5.3 shows the result of using MCMC to fit a 10-component mixture of normals to the mixing distribution. The weights in the Gaussian mixture were assigned three different symmetric Dirichlet priors, with concentration parameter $\alpha \in \{0.01, 1, 100\}$. Panels 1-3 in Figure 5.3 show the posterior mean and posterior 95% credible envelopes for f_0; these settings span a wide range of expected degrees of smoothness for f_0, and they yield a correspondingly wide range of posterior estimates. Comparing figures 5.2 and 5.3, we see that the deconvolution path spans

essentially the entire range of plausible posterior estimates for f_0 arising under any of the concentration parameters. In contrast, Panels 4-7 in Figure show that the kernel estimates are either overly smooth or wiggly.

5.5 Theoretical properties

In this section we establish some important theoretical properties of our estimators by thinking of them as approximations to sieves problems. We start by showing consistency of the mixing density in L1 norm. We do not provide convergence rates since, unlike the kernel estimator from [56] and the predictive recursion from [107], our method cannot be expressed in analytical form. This is out of the scope of our work, but we do provide evidence in the later sections that our estimator can outperform existing non-parametrics methods.

Throughout we consider $k \in \mathbb{N} - \{0\}$ and $q > 0$ to be fixed. We also denote by $\mathcal{P}$ the set of densities in $\mathbb{R}$, thus $\mathcal{P} := \{f : \int_{\mathbb{R}} f(\mu)d\mu = 1; f \geq 0\}$, where $d\mu$ denotes Lebesgue measure. Moreover, given any non-negative function f we say that $b \in T_f$ if

$$\max\left(\|f\|_\infty, \|\left(\log f\right)^{(k+1)}\|_\infty, |\left(\log f\right)^{(k)}(0)|, \ldots, |\left(\log f\right)(0)|\right) \leq b,$$

and $\left(\log f\right)^{(k+1)}$ is b-Lipschitz. Here, given an arbitrary function g, we use the notation $\|\cdot\|_\infty$ to indicate the usual supremum norm on the support of g. Moreover g is called T_m-Lipschitz if it satisfies $|g(x) - g(y)| \leq T_m |x - y|$, for all x and y.

In this section two metrics of interest will be repeatedly used. The first one is the usual ℓ_1 distance $d(f,g) = \int_{\mathbb{R}} |f-g|$. The other metric of interest will be the Hellinger distance whose square is given as $H^2(f,g) := \int_{\mathbb{R}} |\sqrt{f}(\mu) - \sqrt{g}(\mu)|^2 d\mu$. We also use the notation

$$D_{\mathrm{KL}}(f|g) = \int f(\mu) \log(f(\mu)/g(\mu)) d\mu.$$

Finally, for $q \in \mathbb{N}$, we define the functional $J_{k,q}$ which will be a generalization of the usual total variation. We set $J_{k,q}(f) := \int_{\mathbb{R}} |f^{(k+1)}(\mu)|^q d\mu$.

Next we state some assumptions for our first consistency result. Our approach is to consider the objective function in (5.3) restricted to a smaller domain than that of its original formulation. This will then allows to prove that the new problem is not ill defined and also its solutions enjoy asymptotic properties of convergence towards the true mixing density. We refer to [62] for a general perspective on sieves.

Assumptions and definitions Let A be a set of functions that satisfies the following.

Assumption 1. Any function $f \in A$ satisfies that $f \in \mathcal{P}$, $f > 0$, $J_{k,q}(\log f) < \infty$, and there exists a constant $t_f \in T_f$.

Assumption 2. For all $m \in \mathbb{N}$, the exists a set $S_m \subset A$ and constants $T_m, K_m > 0$ such that for all $f \in S_m$ it holds that $t_f = T_m$ and $J_{k,q}(\log f) \leq K_m$. Moreover, for all m, the set S_m induces a tight set of of probability measures in $(\mathbb{R}, \mathcal{B}(\mathbb{R}))$ satisfying $S_m \subset S_{m+1}$. In addition, $\cup_m S_m$ is dense in A with respect to the metric d.

Assumption 3. Data model: we assume that $y_1, \ldots, y_n$ are independent draws from the density $\phi * f_0$, $f_0 \in A$, with ϕ being an arbitrary density function satisfying $\max(\|\phi\|_\infty, \|\phi'\|_\infty) < \infty$ and $\int_{\mathbb{R}} \log(\phi * f_0(\mu))\, \phi * f_0(\mu) d\mu < \infty$.

Assumption 4. The set

$$A_m = \left\{ \alpha \in S_m \,:\, D_{\mathrm{KL}}(\phi * f_0 || \phi * \alpha) = \inf_{\beta \in S_m} D_{\mathrm{KL}}(\phi * f_0 || \phi * \beta) \right\},$$

satisfies $d(f_0, \alpha) \to 0$ as $m \to \infty$ for all $\alpha \in A_m$, where the convergence is uniform in A_m.

Assumption 5. We assume that the $y_1, \ldots, y_n$ are binned into D_n different intervals with frequency counts $\{x_j\}_{j=1,\ldots,D_n}$ such that $n^{-1}\|x\|_\infty \to 0$ $a.s.$, and we denote by ξ_j an arbitrary point in interval j. Note that this trivially holds for the case where $D_n = n$ and $x_j = 1$f for all j.

Assumption 6. There exists $f_m \in A_m$ such that

$$\sum_{j=1}^{D_n} \frac{x_j}{n} \log(\phi * f_m(\xi_j)) \to \int_{\mathbb{R}} \log(\phi * f_m(\xi))\, \phi * f_0(\xi) d\xi \text{ a.s. as } n \to \infty.$$

If the $x_j = 1$ and $\xi_j = y_j$ for all $j = 1, \ldots,$ this condition can be disregarded.

Assumptions (1)-(3) are natural for the original variational problem proposed earlier. The Lipschitz condition, the bounds on the behavior of the functions at zero, and the tightness of distributions are merely used to ensure that the sieves will indeed be compact sets with respect to the metric d. Moreover, Assumption (4) tell us that the sieves S_m are rich enough to approximate the true mixing density sufficiently well. The last two assumptions can be disregarded when the counts in the bins are all one.

We are now ready to state our first consistency result. Its proof generalizes ideas from Theorem 1 in [62].

Theorem 5.5.1. *If Assumptions (1-6) hold, then, the problem*

$$\underset{f \in S_m}{\textit{minimize}} \quad -\sum_{j=1}^{D_n} x_j \log\left(\phi * f\right)(\xi_j)$$

has solution set $M_m^n \neq \emptyset$*. Moreover, for any sequence* m_n *increasing slowly enough it holds that*

$$\underset{\beta \in M_{m_n}^n}{\textit{sup}} \; d\left(\beta, f_0\right) \to 0 \;\; \textit{a.s.}$$

Proof. See Appendix D.1.1. □

In Theorem 5.5.1, the sequence T_{m_n} is arbitrary and can grow as fast as desired. Moreover, the a.s statement is on the probability space

$$\left(\mathbb{R}^\infty, \mathcal{F}, F_0 \times F_0 \times F_0, \ldots\right),$$

with F_0 the measure on $(\mathbb{R}, \mathcal{B}(\mathbb{R}))$ induced by $\phi * f_0$, and with $\mathcal{F}$ the completion of $\mathcal{B}\left(\mathbb{R}\right)^\infty$.

5.6 Experiments

5.6.1 Mixing density estimation

In this section we show the potential gain given by our penalized approaches. We start by considering the task of recovering the true mixing distribution. We evaluate the performance of our methods described in

Figure 5.4: The first panel shows a histogram of observed data $\{y_i\}_{i=1}^n$ for our first example, and the L1-D marginal density estimate plotted on top of the histogram. Here the data has been generated as $y_i \sim N(\mu_i, 1)$ where μ_i is a draw from the mixing density. The second panel shows, for this same example, the histogram of $\{\mu_i\}_{i=1}^n$ (unobserved draws from the mixing density) and the L1-D estimate of the mixing density plotted on top of it. Panels 3-6 show the respective cases of Examples 2 and 3. The last two panels show the corresponding plots for the L2-D solution and Example 4.

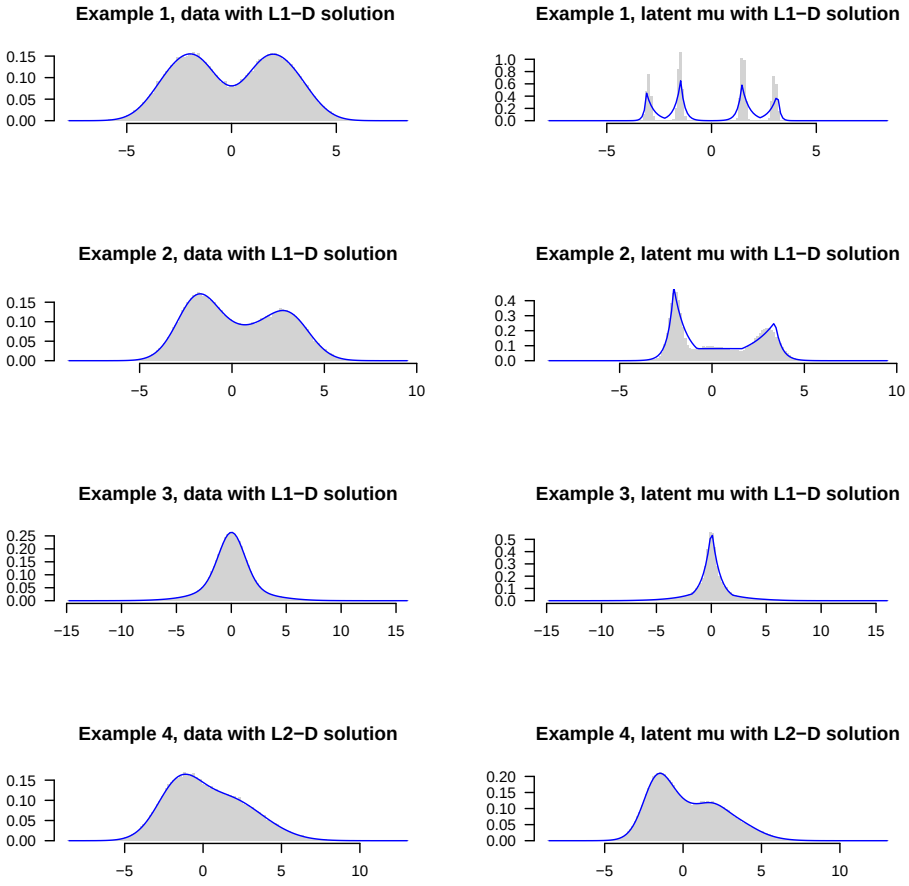
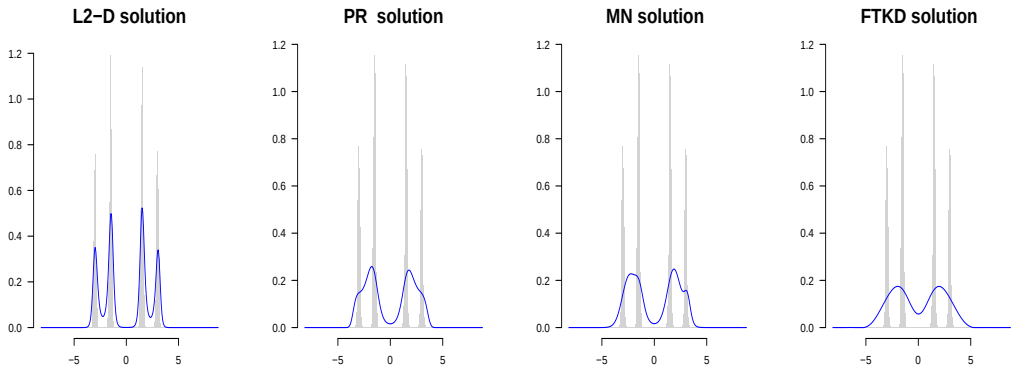


Figure 5.5: For the mixing density illustrated in Example 1 of Figure 5.4 we show the estimated mixing densities of different methods. The top two panels correspond to the estimated mixing densities using L2-D and PR algorithms along with latent μ. Bottom two panels show the estimated density using MN and FTKD both with the latent μ. For all four panels $n = 10^5$



Section 5.3 which we call L1-deconvolution (L1-D) and L2-deconvolution (L2-D) depending on the regularization penalty used in the estimation. As competitors we consider a mixture of normals model (MN), the predictive recursion algorithm (PR) from [107], the Fourier transform kernel deconvolution method (FTKD) from [56], and the "g-modeling" method from [52] (g-M). Our comparisons are based on four examples which are shown in Figure 5.4. These examples are intended to illustrated the performance under different scenarios involving smooth and sharp densities. Next we describe the simulation setting and as well as the implementation details of the competing methods.

As a flexible Bayesian model we decided to use a prior for the mixing density based on a mixture of 10 normals (MN). Here, the weights of the

mixture components are drawn from a Dirichlet prior with concentration parameter 1. This is done in order to have a uniform prior on the simplex. For the locations of the mixture we consider non-informative priors given as $N(0, 10^2)$ while for the variances of the mixture components we place a inverse gamma prior with shape parameter 0.01 and rate 0.01. The complete model can be then thought as a weak limit approximation to the Dirichlet process, [72]. Also, Gibss sampling is accomplished straightforwardly by introducing a data augmentation with a variable z_i indicating the component to which μ_i belong.

The next competing model is the predictive recursion algorithm from [107] for which we choose the weights w_i as in [103], close to the limit of the upper bound of the convergence rate for PR given in [143]. Moreover we average the PR estimator over 10 different permutations of the input data in order to obtain a smaller expected error [143].

On the other hand, for the Fourier transform kernel deconvolution method, we consider different choices of bandwidth: the rule of thumb from [56], the plug in bandwidth from [132], and the 2-stage plug-in bandwidth from [38]. Our estimates are obtained using the R package fDKDE available at `http://www.ms.unimelb.edu.au/~aurored/links.html`, which addresses the main concerns associated with the R package decon, see [37].

For the final competitor, the "g-modeling" approach from [52], we use the newly released R package deconvolveR.

Table 5.1: Mean squared error (MSE) between the true and estimated mixing densities, averaging over 100 Monte Carlo simulations, for different methods given samples from density Example 1. The acronyms here are given the text. The MSE is multiplied by 10^2 and reported over two intervals containing 95% and 99% of the mass of the mixing density.

n	MN 95%	PR 95%	L2-D 95%	L1-D 95%	FTKD 95%	g-M 95%
2000	9.47	9.12	9.39	9.69	**8.89**	9.27
10000	8.43	8.72	8.64	**7.44**	8.87	9.22
25000	8.34	8.46	7.27	**5.54**	8.88	9.32
50000	8.21	8.23	5.80	**4.15**	8.85	9.40
100000	8.34	8.05	4.79	**3.38**	8.69	9.50
n	MN 99%	PR 99%	L2-D 99%	L1-D 99%	FTKD 99%	g-M 99%
2000	9.26	8.91	9.18	9.48	**8.89**	9.01
10000	8.24	8.52	8.44	**7.28**	8.87	9.00
25000	8.15	8.27	7.13	**5.43**	8.16	9.09
50000	8.03	8.04	5.71	**4.09**	8.66	9.18
100000	8.14	7.86	4.69	**3.35**	8.49	9.28

We now state the simulation setting for recovering the mixing density. Given the densities from Figure 5.4, we consider varying the number of samples n and for each fixed n we run 100 Monte Carlo simulations. Moreover, for our methods we set D, the number of evenly space points in the grid, to 250. See the appendix for a sensitivity example of this parameter.

The results on Table 5.1 illustrate a clear advantage of our penalized likelihood approaches over MN, PR and FTKD which seems even more significant for larger samples size. The estimated mixing density by L1-D is shown in Figure 5.4 where we can clearly see that L1-D can capture the peaks of the unknown mixing density. Moreover, Figure 5.5 shows that L2-

Table 5.2: Mean squared error (MSE) between the true and estimated mixing densities, averaging over 100 Monte Carlo simulations, for different methods given samples from density Example 2. The acronyms here are given the text. The MSE is multiplied by 10^3 and reported over two intervals containing 95% and 99% of the mass of the mixing density.

n	MN	PR	L2-D	L1-D	FTKD	g-M
	95%	95%	95%	95%	95%	95%
2000	6.20	2.54	2.74	**2.38**	6.07	8.23
10000	3.45	1.75	**1.60**	1.68	5.98	5.82
25000	2.31	1.46	**1.19**	1.35	5.99	5.01
50000	1.24	1.28	**0.89**	1.18	5.89	4.68
100000	0.78	1.07	**0.74**	0.87	4.97	4.03
n	MN	PR	L2-D	L1-D	FTKD	g-M
	99%	99%	99%	99%	99%	99%
2000	5.47	2.37	2.49	**2.13**	5.86	7.06
10000	3.05	1.60	**1.46**	1.49	5.76	5.45
25000	2.20	1.35	**1.09**	1.19	5.70	4.77
50000	1.10	1.17	**0.81**	1.05	5.66	4.39
100000	0.69	0.98	**0.67**	0.77	4.85	3.85

D can also capture the structure of the true density. In contrast, MN, PR and FTKD all fail to provide reliable estimators.

For our example density 2, we observe from Table 5.2 that in general L2-D and L1-D offer the best performance. In the case of example 3 (see Table 5.3), we observe that the L1-D again provides better results than the competitors in all the scenarios of sample sizes considered. Even with only 10000 samples L1-D is closer to the true density than all the other methods with more samples. Moreover, L2-D performs much better than PR and FTKD. Also, L2-D seems to be a clear competitor to MN. In the final example density 4, we observe that L2-D is the best method in all the scenarios

Table 5.3: Mean squared error (MSE) between the true and estimated mixing densities, averaging over 100 Monte Carlo simulations, for different methods given samples from density Example 3. The acronyms here are given the text. The MSE is multiplied by 10^3 and reported over two intervals containing 95% and 99% of the mass of the mixing density.

n	MN	PR	L2-D	L1-D	FTKD	g-M
	95%	95%	95%	95%	95%	95%
2000	5.28	1.83	2.09	**0.96**	4.95	7.16
10000	3.06	1.38	1.46	**0.61**	4.86	1.45
25000	1.51	1.16	1.18	**0.47**	4.61	1.18
50000	0.95	1.06	1.00	**0.42**	3.77	1.13
100000	0.72	0.95	0.86	**0.38**	3.48	2.42
n	MN	PR	L2-D	L1-D	FTKD	g-M
	99%	99%	99%	99%	99%	99%
2000	3.45	1.21	1.37	**0.63**	3.25	4.54
10000	1.99	0.90	0.95	**0.40**	3.18	9.49
25000	0.99	0.72	0.77	**0.31**	3.01	7.75
50000	0.62	0.70	0.66	**0.28**	2.47	9.11
100000	0.47	0.62	0.56	**0.25**	2.29	1.58

considered as shown in Table 5.4.

Overall, we have shown that for estimating the mixing density, L1-D and L2-D can perform well under different settings, even when other methods exhibit notable deficiencies. The advantage is amplified by the fact that both of our methods are less computationally intensive that MN, with L2-D requiring around 40 seconds to handle problems with $D = 250$, and L1-D under the same problem conditions typically requires around 5 minutes for a full solution path across 50 values of the tuning parameter.

Table 5.4: Mean squared error (MSE) between the true and estimated mixing densities, averaging over 100 Monte Carlo simulations, for different methods given samples from density Example 4. The acronyms here are given the text. The MSE is multiplied by 10^4 and reported over two intervals containing 95% and 99% of the mass of the mixing density.

n	MN	PR	L2-D	L1-D	FTKD	g-M
	95%	95%	95%	95%	95%	95%
2000	20.6	4.75	**1.82**	3.48	3.25	7.06
10000	7.64	1.89	**0.65**	1.93	2.87	4.20
25000	2.04	1.10	**0.48**	2.19	2.57	2.34
50000	1.03	0.69	**0.36**	1.20	2.02	1.95
100000	0.50	0.55	**0.39**	0.90	1.36	1.49
n	MN	PR	L2-D	L1-D	FTKD	g-M
	99%	99%	99%	99%	99%	99%
2000	16.8	4.03	**1.15**	2.88	3.00	5.88
10000	6.23	1.60	**0.53**	1.60	2.67	3.53
25000	1.67	0.93	**0.39**	1.80	2.37	1.95
50000	0.85	0.58	**0.30**	1.00	1.86	1.62
100000	0.40	0.46	**0.32**	0.85	1.25	1.22

5.6.2 Normal means estimation

After evaluating our proposed methodology for the task of estimating the mixing density, we now, for the case of standard normal kernel, focus on the estimation of the normals means $\{\mu_i\}$. For this, we consider comparisons using the best four among the methods used before in addition to other procedures that we briefly discuss next.

As it is well known (see for instance [51] for description and references), assuming that the marginal density is known, one can use Tweedie's formula to estimate $\{\mu_i\}$. For all the methods here this is the approach that we take, except for MN in which case we use the posterior means resulting

from Gibss sampling inference. For the methods depending on grid estimator, the number of bins is set to 250.

For the method of [51], we set to 5 the degree of the polynomial approximation to the logarithm of the marginal true density (we found larger values to be less numerically stable). The Poisson surrogate model is then fit in R using the command glm. We also compare against the general maximum likelihood empirical-Bayes estimator (GMLEB) from [73], which is a discretized version of the original Kiefer–Wolfowitz estimator. For our comparisons we use the algorithm proposed in [84] based on an interior point method algorithm (GMLEBIP). We use the R package REBayes in order to obtain this estimator ([81]). On the other hand, for the shape constrained (SC) estimator from [84], we rely on a binned count approach based on a weighted likelihood using R code proived by the authors. Moreover, we consider the estimator from [18] using the default choice of bandwidth $h_n = (\log n)^{-1/2}$, which we refer to as BG. The finally competitor is the nonlinear projection (NLP) estimator from [147].

From Table 5.5 it is clear that the best methods for example 1 are L1-D, L2-D, GMLEBIP, and NLP. Moreover, it is not surprising that GMLEBIP provides good estimates given that the true mixing density has mixture components that have small variance.

For example 2, we can see from Table 5.6 that again L2-D and L1-D provide competitive estimates. The other suitable methods for this example seem to be PR and GMLEBIP. With slightly worse estimates MN, BG

Table 5.5: Mean squared error, of the normal means estimates, times 100 , averaging over 100 Monte Carlo simulations, for different methods given samples from example 1.

n	L2-D	L1-D	PR	MN	Efron	GMLEBIP	SC	BG	NLP
2000	64.31	64.29	64.16	67.50	70.27	64.48	68.24	65.57	**64.11**
10000	63.89	63.68	63.86	**63.18**	70.00	63.80	65.56	64.06	63.27
25000	63.52	**63.37**	63.69	63.84	69.96	63.39	64.66	63.65	63.60
50000	63.27	**63.21**	63.55	65.20	69.85	63.23	64.15	63.44	63.26
100000	63.27	63.23	63.59	63.79	69.89	63.21	63.86	63.39	**63.18**

Table 5.6: Mean squared error, of the normal means estimates, times 100, averaging over 100 Monte Carlo simulations, for different methods given samples from example 2.

n	L2-D	L1-D	PR	MN	Efron	GMLEBIP	SC	BG	NLP
2000	65.42	65.46	65.36	**64.33**	69.97	66.20	69.60	66.99	65.75
10000	**64.98**	65.06	65.08	65.66	69.75	65.29	67.10	65.54	65.95
25000	65.19	**65.08**	65.32	65.21	69.94	65.12	66.42	65.49	65.09
50000	**64.99**	65.08	65.13	65.44	69.93	65.03	65.97	65.19	65.24
100000	65.02	**64.95**	65.14	65.03	69.84	65.02	65.69	65.14	64.96

and SC provide results that are still competitive, with SC being particularly attractive given its computational speed to provide solutions.

Finally, for examples 3 and 4 we can see in Tables 5.7 and 5.8 respectively that L1-D and L2-D are the best or among the best methods in terms of mean squared distance when recovering the unknown means μ_i. Table 5.8 also suggests that Efron's estimator is more suitable when the true mixing density is very smooth with no sharp peaks.

Table 5.7: Mean squared error, of the normal means estimates, times 100, averaging over 100 Monte Carlo simulations, for different methods given samples from example 3.

n	L2-D	L1-D	PR	MN	Efron	GMLEBIP	SC	BG	NLP
2000	**64.99**	64.96	65.41	69.05	70.54	65.74	69.70	66.99	65.77
10000	64.73	64.76	64.96	**64.21**	71.34	64.85	66.92	65.36	64.81
25000	**64.52**	64.57	64.75	64.97	71.42	64.65	66.62	64.82	64.62
50000	**64.51**	64.61	64.73	65.38	71.52	64.64	66.60	64.67	64.57
100000	64.54	**64.41**	64.76	64.54	71.96	64.56	65.17	64.62	64.46

Table 5.8: Mean squared error, of the normal means estimates, times 100, averaging over 100 Monte Carlo simulations, for different methods given samples from example 4.

n	L2-D	L1-D	PR	MN	Efron	GMLEBIP	SC	BG	NLP
2000	79.63	80.20	79.89	**78.68**	80.00	80.97	85.47	81.58	80.01
10000	**79.32**	79.35	79.42	79.34	79.99	79.74	82.18	79.89	79.64
25000	79.39	79.31	79.48	**78.79**	79.96	79.30	80.98	79.65	79.39
50000	**79.21**	79.25	79.29	79.85	79.82	79.40	80.58	79.36	79.39
100000	79.29	**79.22**	79.37	79.51	79.91	79.30	80.15	79.37	79.36

5.7 Discussion

In many problems in statistics and machine learning, we observe a blurred version of an unknown mixture distribution which we would like to recover via deconvolution. The main challenge is to find an approach that is computationally fast but still possesses nice statistical guarantees in the form of rates of convergence. We propose a two-step "bin-and-smooth" procedure that achieves both of these goals. This reduces the deconvolution problem to a Poisson-regularized model would can be solved either

via standard methods for smooth optimization, or with a fast version of the alternating-direction method of multipliers (ADMM). Our approach reduces the computational cost compared to a fully Bayesian method and yields a full deconvolution path to illustrate the sensitivity of our solution to the specification of the amount of regularization. We provide theoretical guarantees for our procedure. In particular, under suitable regularity conditions, we establish the almost-sure convergence of our estimator towards the mixing density. We also characterize convergence rates for recovery of marginal density and illustrate the type of sensitivity analysis that can be performed in our framework.

There are a number of directions for future inquiry, including multivariate extensions and extensions to multiple hypothesis testing. These are active areas of current research.

Chapter 6

Concluding remarks

6.1 Summary

This thesis has presented novel contributions to the statistical area of regularized likelihood estimation, particularly through new methodologies using ideas of total variation and its generalizations. Two important aspects have been emphasized throughly: practicality of algorithms, and statistical accuracy.

One of the main contributions of this work is to prove universal convergence rates for the fused lasso, regardless of the graph structure. This has been done in the form of an upper bound on the mean squared error performance of the fused lasso, as described in Chapter 2. Perhaps strikingly, this has been showed with a previously unknown connection between the widely used depth first search algorithm and the notion of total variation on graphs. This has led us to provide a simple two–step estimator: first run the DFS algorithm to obtain an ordering of nodes, and then use such ordering to run the 1d fused lasso. While this might seem unintuitive at first sigh, we have proven that such simple procedure actually achieves the same universal upper bound that holds for the fused lasso, which we also

proved.

In Chapter 3, we have presented a general framework for smooth low rank tensor decompositions, which is based on trend filtering, the higher order version of the fused lasso. Importantly, our methodology is amenable for interpretation and prediction purposes as illustrated y the real data examples we have considered. Another attractive aspect in our methods has been the substitution of a non-convex optimization problem by a simple algorithm that only involves solving convex problems, exploiting state of the art advances in trend filtering regression.

Finally, we have shown how the ideas of trend filtering regression can be used to effectively perform non-parametric deconvolution and density estimation. A key insight behind our construction is how the idea of binning the data can be combined with new optimization algorithms to lead to low cost algorithms. In a addition to being computational attractive, a more important feature of our deconvolution and density estimators is their adaptivity to functions with different degrees of smoothness in their domain. Thus, our proposed approaches offer a useful combination of local adaptivity and low computational cost.

6.2 Future work

6.2.1 DFS fused lasso

In Section 2.6 we have presented many different lines of work for the chaining idea introduced in Chapter 2. In all these discussions, we proposed

to consider somehow running the DFS algorithm to get an ordering and then run a optimization algorithm, depending on the context, using such ordering. It is left for future work to actually validate whether such idea would work in practice for Energy minimization and the Potts model.

Another possibility for DFS fused lasso is the idea of boosting. Thus, somehow combining different iterations of DFS fused lasso, hence going beyond averaging.

Finally, a further direction of work could be to understand the convergence rates of trend filtering, the higher order version of the fused lasso, on generic graphs. This seems to be plausible, given the optimistic results we have shown in this thesis for the fused lasso.

6.2.2 Tensor decompositions

In Chapter 3 we have been interested in Parafac models for tensor decompositions, which are special cases of general Tucker models. A penalized Tucker model was proposed in [25] in which the goal was to maximize, with respect to $U^{(n)} \in \mathbb{R}^{I_n \times J_n}$, $n = 1, \ldots, N$, the cost function

$$D_F\left(\underline{Y} \| G, \{U\}\right) = \| \underline{Y} - G \times \{U\} \|_F^2 + \sum_n \alpha_n C_n \left(U^{(n)}\right), \tag{6.1}$$

where $\underline{Y}$ is a given data tensor, and $C_1, \ldots, C_n$ are penalties on $U^{(1)}, \ldots, U^{(n)}$. Here, $\alpha_1, \ldots, \alpha_n$ are positive tuning parameters. It is a natural extension to the framework in Chapter 3 to consider decompositions similar to (6.1). This is something that we leave for future work.

6.2.3 Density estimation and deconvolution

One natural extension of our histogram trend filtering density estimator is to consider the problem of non-parametric conditional density estimation. For instance, for the case univariate random variables this is immediate from Section 4.7 .

A perhaps more challenging extension would be to consider the task of spatial density estimation, as in the context of [134]. This would require to construct a penalty that not only encourage smooth densities but allows to incorporate spatial information across different sites of the spatial network.

On the other hand, for deconvolution there are some important questions that we have not explored and that would be of interest in practice and also from a theoretical point of view. These are briefly described next.

In many biological applications one is interested in multiple testing problems, where samples are tested to be draws from a theoretical null versus the alternative of being from some other distribution, see [50]. However, in practice, [50] points that the theoretical null, which is usually assumed to be standard Gaussian, needs to be estimated. Thus, we can envision to construct a deconvolution estimator, inspired by Chapter 5, that jointly estimates the null and alternatives in the two group model. This would then allow us to preform false discovery rate. Some preliminary results are encouragingly suggesting that this could be a fruitful research problem that is left for future work.

Finally, we have empirically shown in Section 5.6.2 that our deconvolution estimator can be useful for the normal means estimation problem. However, we have not characterized in any mathematical way the performance of our approach within that context. It is out of the scope of this thesis to answer such question.

Appendices

Appendix A

Proofs for Chapter 2

A.1 Derivation of (2.12) from Theorem 3 in [149]

We first establish a result on the exact form for the inverse of (an augmented version of) the edge incidence matrix of a generic tree $T = (V, E_T)$, where, recall $V = \{1, \ldots, n\}$. Without a loss of generality, we may assume that the root of T is at node 1. For $m \leq n$, we define a path in T, of length m, to be a sequence $p_1, \ldots, p_m$ such that $\{p_r, p_{r+1}\} \in E_T$ for each $r = 1, \ldots, m-1$. We allow for the possibility that $m = 1$, in which case the path has just one node. For any $j, k, \ell = 1, \ldots, n$, we say that j is on the path from k to ℓ if there exists a path $p_1, \ldots, p_m$ such that $p_1 = k$, $p_m = \ell$ and $p_r = j$ for some $r = 1, \ldots, m$. For each node $i = 2, \ldots, n$ (each node other than the root), we define its parent $p(i)$ to be the node connected to i which is on the path from the root to i.

We can also assume without a loss of generality that for each $i = 2, \ldots, n$, the $(i-1)$st row of ∇_T corresponds to the edge $\{p(i), i\}$, and thus we can write

$$(\nabla_T)_{i-1,j} = \begin{cases} -1 & \text{if } j = p(i), \\ 1 & \text{if } j = i, \\ 0 & \text{if } j \in \{1, \ldots, n\} \setminus \{i, p(i)\}. \end{cases}$$

for each $j = 1, \ldots, n$. The next lemma describes the inverse of ∇_T, in the appropriate sense.

Lemma A.1.1. *Let $e_1 = (1, 0, \ldots, 0) \in \mathbb{R}^n$, and define the matrix $A_T \in \mathbb{R}^{n \times n}$ by*

$$(A_T)_{i,j} = \begin{cases} 1 & \text{if } j \text{ is on the path from the root to } i, \\ 0 & \text{otherwise}, \end{cases} \tag{A.1}$$

for each $i, j = 1, \ldots, n$. Then

$$A_T = \begin{pmatrix} e_1^\top \\ \nabla_T \end{pmatrix}^{-1}.$$

Proof. We will prove that the product

$$B = \begin{pmatrix} e_1^\top \\ \nabla_T \end{pmatrix} A_T$$

is the identity. As the root of T corresponds to node 1, we have that by definition of A_T that its first column is

$$(A_T)_{\cdot,1} = (1, \ldots, 1),$$

which implies that the first column of B is

$$B_{\cdot,1} = e_1.$$

Moreover, by definition of A_T, its first row is

$$(A_T)_{1,\cdot} = e_1^\top,$$

which implies that the first row of B is

$$B_{1,\cdot} = e_1^\top.$$

Let us now assume that i, j are each not the root. We proceed to consider three cases.

Case 1. Let $j \neq i$, and j be on the path from the root to i. Then j is also on the path from the root to $p(i)$. This implies that

$$B_{ij} = \begin{pmatrix} e_1^\top \\ \nabla_T \end{pmatrix}_{i,\cdot} (A_T)_{\cdot,j} = (\nabla_T)_{i-1,\cdot}(A_T)_{\cdot,j} = 1 - 1 = 0.$$

Case 2. Let $j \neq i$, and j not be on the path from the root to i. Then j is not on the path from the root to $p(i)$, which implies that

$$B_{ij} = \begin{pmatrix} e_1^\top \\ \nabla_T \end{pmatrix}_{i,\cdot} (A_T)_{\cdot,j} = (\nabla_T)_{i-1,\cdot}(A_T)_{\cdot,j} = 0 - 0 = 0.$$

Case 3. Let $j = i$. Then j is on the path from the root to i, and j is not on the path from the root to $p(i)$. Hence,

$$B_{ij} = \begin{pmatrix} e_1^\top \\ \nabla_T \end{pmatrix}_{i,\cdot} (A_T)_{\cdot,j} = (\nabla_T)_{i-1,\cdot}(A_T)_{\cdot,j} = -1 \cdot 0 + 1 \cdot 1 = 1.$$

Assembling these three cases, we have shown that $B = I$, completing the proof. □

We now establish (2.12).

Proof of (2.12). The proof of Theorem 3 in [149] proceeds as in standard basic inequality arguments for the lasso, and arrives at the step

$$\|\Pi^\perp(\hat{\theta}_G - \theta_0)\|_2^2 \leq 2\epsilon^\top \Pi^\perp(\hat{\theta}_G - \theta_0) + 2\lambda\|\nabla_G \theta_0\|_1 - 2\lambda\|\nabla_G \hat{\theta}_G\|_1,$$

where $\Pi^{\perp}$ is the projection matrix onto the space $\mathbf{1}^{\perp}$, i.e., the linear space of all vectors orthogonal to the vector $\mathbf{1} = (1, \ldots, 1) \in \mathbb{R}^n$ of all 1s. The proof in [149] uses the identity $\Pi^{\perp} = \nabla_G^{\dagger} \nabla_G$, where $\nabla_G^{\dagger}$ denotes the pseudoinverse of ∇_G. However, notice that we may also write $\Pi^{\perp} = \nabla_T^{\dagger} \nabla_T$ for any spanning tree T of G. Then, exactly the same arguments as in [149] produce the MSE bound

$$\|\hat{\theta}_G - \theta_0\|_n^2 = O_{\mathbb{P}}\left(\frac{M(\nabla_T)\sqrt{\log n}}{n} \|\nabla_G \theta_0\|_1 \right),$$

where $M(\nabla_T)$ is the maximum ℓ_2 norm among the columns of $\nabla_T^{\dagger}$. We show below, using Lemma A.1.1, that $M(\nabla_T) \leq \sqrt{n}$, and this gives the desired MSE rate.

For any $b \in \mathbb{R}^{n-1}$, we may characterize $\nabla_T^{\dagger} b$ as the unique solution $x \in \mathbb{R}^n$ to the linear system

$$\nabla_T x = b,$$

such that $\mathbf{1}^{\top} x = 0$, i.e., the unique solution to the linear system

$$\begin{pmatrix} e_1^{\top} \\ \nabla_T \end{pmatrix} x = \begin{pmatrix} a \\ b \end{pmatrix},$$

for a value of $a \in \mathbb{R}$ such that $\mathbf{1}^{\top} x = 0$. By Lemma A.1.1, we may write

$$x = A_T \begin{pmatrix} a \\ b \end{pmatrix},$$

so that the constraint $0 = \mathbf{1}^{\top} x = na + \mathbf{1}^{\top} (A_T)_{\cdot, 2:n} b$ gives $a = -(\mathbf{1}/n)^{\top} (A_T)_{\cdot, 2:n} b$, and

$$x = (I - \mathbf{1}\mathbf{1}^{\top}/n)(A_T)_{\cdot, 2:n} b.$$

Evaluating this across $b = e_1, \ldots, e_n$, we find that the maximum ℓ_2 norm of columns of $\nabla_T^\dagger$ is bounded by the maximum ℓ_2 norm of columns of $(A_T)_{\cdot,2:n}$, which, from the definition in (A.1), is at most $\sqrt{n}$. □

A.2 Proof of Theorem 2.3.3

We first present two preliminary lemmas.

Lemma A.2.1. *Let $S_1, \ldots, S_m$ be a partition of the nodes of G such that the total number of edges with ends in distinct elements of the partition is at most s. Let $k \leq \min_{i=1,\ldots,m} |S_i|$. Then*

$$\inf_{\hat{\theta}} \sup_{\theta_0 \in \mathrm{BV}_G(t)} \mathbb{E}\|\hat{\theta} - \theta_0\|_2^2 \geq \frac{kmt^2}{4\sigma^2 s^2} \exp\left(-\frac{kt^2}{\sigma^2 s^2}\right).$$

Proof. For each $\eta \in \{-1, 1\}^m$, define

$$\theta_\eta = \frac{\delta}{2} \sum_{i=1}^m \eta_i \frac{1_{S_i}}{\sqrt{|S_i|}},$$

where $\delta > 0$ will be specified shortly. Also define the class

$$\mathcal{P} = \{N(\theta_\eta, \sigma^2 I) : \eta \in \{-1, 1\}^m\}.$$

Note that $\|\nabla_G \theta_\eta\|_1 \leq \delta s/\sqrt{k}$, so to embed $\mathcal{P}$ into the class $\{N(\theta, \sigma^2 I) : \theta \in \mathrm{BV}_G(t)\}$, we set $\delta = t\sqrt{k}/s$.

Let $\eta, \eta' \in \{-1, 1\}^m$ differ in only one coordinate. Then the KL divergence between the corresponding induced measures in $\mathcal{P}$ is $\|\theta_\eta - \theta_{\eta'}\|_2^2/\sigma^2 \leq \delta^2/\sigma^2$. Hence by Assouad's Lemma [152], and a well-known lower bound on the

affinity between probability measures in terms of KL divergence,

$$\inf_{\hat{\theta}} \sup_{\theta_0 \in \mathrm{BV}_G(t)} \mathbb{E}\|\hat{\theta} - \theta_0\|_2^2 \geq \frac{\delta^2 m}{4\sigma^2} \exp\bigg(-\frac{\delta^2}{\sigma^2}\bigg).$$

The result follows by plugging in the specified value for δ. □

Lemma A.2.2. *Let G be a tree with maximum degree $d_{\max}$, and $k \in \{1, \ldots, n\}$ be arbitrary. Then there exists a partition as in Lemma A.2.1, $s = m - 1$, and*

$$k \leq \min_{i=1,\ldots,m} |S_i| \leq k(d_{\max} + 1).$$

Proof. Our proof proceeds inductively. We begin by constructing S_1', the smallest subtree among all those having size at least k, and generated by a cut of size 1 (i.e., separated from the graph by the removal of 1 edge). Note that $|S_1'| \leq kd_{\max}$, because if not then S_1' has at least k internal nodes, and we can remove its root to produce another subtree whose size is smaller but still at least k.

For the inductive step, assume $S_1', \ldots, S_\ell'$ have been constructed. We consider two cases. (For a subgraph G' of G, we denote by $G - G'$ the complement subgraph, given by removing all nodes in G', and all edges incident to a node in G'.)

Case 1. If $|G - \cup_{i=1}^{\ell} S_i'| > k$, then we construct $S_{\ell+1}'$, the smallest subtree of $G - \cup_{i=1}^{l} S_i'$ among all those having size at least k, and generated by a cut of size 1. As before, we obtain that $|S_{\ell+1}'| \leq kd_{\max}$.

Case 2. If $|G - \cup_{i=1}^{\ell} S_i'| \leq k$, then the process is stopped. We define $S_i = S_i'$, $i = 1, \ldots, \ell - 1$, as well as $S_\ell = S_\ell' \cup (G - \cup_{i=1}^{\ell} S_i')$. With $m = \ell$, the result follows. □

We now demonstrate a more precise characterization of the lower bound in Theorem 2.3.3, from which the result in the theorem can be derived.

Theorem A.2.3. *Let G be a tree with maximum degree $d_{\max}$. Then*

$$\inf_{\hat{\theta}} \sup_{\theta_0 \in \mathrm{BV}_G(t)} \mathbb{E}\|\hat{\theta} - \theta_0\|_2^2 \geq \frac{t^2}{4e\sigma^2 n} \left(\left(\frac{\sigma n}{2t(d_{\max} + 1)} \right)^{2/3} - 1 \right)^2.$$

Proof. Set $s = m - 1$ and

$$k = \left\lfloor \left(\frac{\sigma n}{2t(d_{\max} + 1)} \right)^{2/3} \right\rfloor.$$

By Lemmas A.2.1 and A.2.2,

$$\begin{aligned}
\inf_{\hat{\theta}} \sup_{\theta_0 \in \mathrm{BV}_G(t)} \mathbb{E}\|\hat{\theta} - \theta_0\|_2^2 &\geq \frac{kmt^2}{4\sigma^2 s^2} \exp\left(- \frac{kt^2}{\sigma^2 s^2} \right) \\
&\geq \frac{kt^2}{4\sigma^2 m} \exp\left(- \frac{kt^2}{\sigma^2 (m-1)^2} \right) \\
&\geq \frac{kt^2}{4\sigma^2 m} \exp\left(- \frac{t^2 k^3 (d_{\max} + 1)^2}{\sigma^2 n^2} \frac{m^2}{(m-1)^2} \right) \\
&\geq \frac{kt^2}{4\sigma^2 n} \exp\left(- \frac{4t^2 k^3 (d_{\max} + 1)^2}{\sigma^2 n^2} \right) \\
&\geq \frac{k^2 t^2}{4\sigma^2 n} \exp(-1).
\end{aligned}$$

In the above, the third line uses $n/m \leq k d_{\max}$ as given by Lemma A.2.2, the fourth line simply uses $m \leq n$ and $m^2/(m-1)^2 \leq 4$ (as $m \geq 2$), and the last

line uses the definition of k. Thus, because

$$k \geq \left(\frac{\sigma n}{2t(d_{\max}+1)} \right)^{2/3} - 1,$$

we have established the desired result. □

A.3 Proof of Theorem 2.4.3

First we establish that, as G is a tree, the number of nodes of degree at most 2 is at least $n/2$. Denote by d_i be the degree of the node i, for each $i = 1, \ldots, n$. Then

$$\begin{array}{rcl} 2(n-1) & = & \sum_{i=1}^{n} d_i = \sum_{i\,:\,d_i \leq 2} d_i + \sum_{i\,:\,d_i \geq 3} d_i \\ & \geq & |\{i : d_i \leq 2\}| + 3|\{i : d_i \geq 3\}| \\ & = & 3n - 2|\{i : d_i \leq 2\}|. \end{array}$$

Hence, rearranging, we find that $|\{i : d_i \leq 2\}| \geq n/2 + 1$.

Let $\mathcal{I} = \{i : d_i \leq 2\}$ so that $|\mathcal{I}| \geq \lceil n/2 \rceil$ and stipulate that $|\mathcal{I}|$ is even without loss of generality. Let k be the largest even number such that $k \leq s/2$. Define

$$\mathcal{B} = \{z \in \mathbb{R}^n : z_{\mathcal{I}} \in \{-1, 0, +1\}^{|\mathcal{I}|},\ z_{\mathcal{I}^c} = 0,\ \|z\|_0 = k\}.$$

Note that by construction $\mathcal{B} \subseteq \mathrm{BD}_G(s)$.

Assume $s \leq n/6$. Then this implies $k/2 \leq n/6 \leq |\mathcal{I}|/3$. By Lemma 4 in [111], there exists $\tilde{\mathcal{B}} \subseteq \mathcal{B}$ such that

$$\log |\tilde{\mathcal{B}}| \geq \frac{k}{2} \log \left(\frac{|\mathcal{I}| - k}{k/2} \right),$$

and $\|z - z'\|_2^2 \geq k/2$ for all $z, z' \in \tilde{\mathcal{B}}$. Defining $\mathcal{B}_0 = 2\delta\tilde{\mathcal{B}}$, for $\delta > 0$ to be specified shortly, we now have $\|z - z'\|_2^2 \geq 2\delta^2 k$ for all $z, z' \in \mathcal{B}_0$.

For $\theta \in \mathcal{B}_0$, let us consider comparing the measure $P_\theta = N(\theta, \sigma^2 I)$ against $P_0 = N(0, \sigma^2 I)$: the KL divergence between these two satisfies

$$K(P_\theta || P_0) = \|\theta\|_2^2/\sigma^2 = 2\delta^2 k/\sigma^2.$$

Let $\delta = \sqrt{\alpha\sigma^2/(2k)\log|\mathcal{B}_0|}$, for a parameter $\alpha < 1/8$ that we will specify later. We have

$$\frac{1}{|\mathcal{B}_0|}\sum_{\theta\in\mathcal{B}_0} K(P_\theta||P_0) \leq \alpha \log|\mathcal{B}_0|.$$

Hence by Theorem 2.5 in [145],

$$\inf_{\hat{\theta}} \sup_{\theta_0 \in \mathrm{BD}_G(s)} \mathbb{P}(\|\hat{\theta} - \theta_0\|_2^2 \geq \delta^2 k) \geq \frac{\sqrt{|\mathcal{B}_0|}}{1+\sqrt{|\mathcal{B}_0|}}\left(1 - 2\alpha - \sqrt{\frac{2\alpha}{\log|\mathcal{B}_0|}}\right). \quad \text{(A.2)}$$

It holds that

$$\delta^2 k = \frac{\alpha\sigma^2}{2}\log|\mathcal{B}_0| \geq \frac{\alpha\sigma^2 k}{4}\log\frac{|\mathcal{I}| - k}{k/2} \geq C\sigma^2 s \log\left(\frac{n}{s}\right),$$

for some constant $C > 0$ depending on α alone. Moreover, the right-hand side in (A.2) can be lower bounded by (say) $1/4$ by taking α to be small enough and assuming n/s is large enough. Thus we have established

$$\inf_{\hat{\theta}} \sup_{\theta_0 \in \mathrm{BD}_G(s)} \mathbb{P}\left(\|\hat{\theta} - \theta_0\|_2^2 \geq C\sigma^2 s \log\left(\frac{n}{s}\right)\right) \geq \frac{1}{4},$$

and the result follows by Markov's inequality.

Appendix B

Proofs and experiments details for Chapter 3

B.1 ADMM algorithm to solve the constrained updates

In this section we discuss how to find the updates for Algorithm 1 from the main document using the ADMM algorithm from [156]. Since these are symmetric we focus on the particular update u^m. In this case the problem is

$$\begin{array}{rcll} u^m & = & \underset{u}{\arg\min} & (-\underline{Y} \times_2 v^{m-1} \times_3 w^{m-1})^T u \\ & & \text{subject to} & \|u\|_2^2 \leq 1\,,\ \|z\|_1 \leq c_u, \\ & & & z = D^u u, \quad (E_u - (D^u)^T D^u)^{1/2} u = \tilde{z}. \end{array} \tag{B.1}$$

We define y as

$$y = \underline{Y} \times_2 v^{m-1} \times_3 w^{m-1}$$

and solve (B.1), using the ADMM algorithm from [156], by considering the iterative updates

$$\begin{array}{rcl} u_{k+1} & = & \underset{\|u\|_2^2 \leq 1}{\arg\min} \left\{ \frac{1}{2}\|y-u\|_2^2 + (2\,\alpha_k - \alpha_{k-1})^T D^u\, u + \frac{\rho}{2}(u-u_k)^T E_u\,(u-u_k) \right\} \\ & = & \frac{\frac{y}{2} - (D^u)^T(\alpha_k - \alpha_{k-1}/2) + \frac{\rho}{2} E_u\, u_k}{\|\frac{y}{2} - (D^u)^T(\alpha_k - \alpha_{k-1}/2) + \frac{\rho}{2} E_u\, u_k\|_2} \\ z_{k+1} & = & \underset{\|z\|_1 \leq c_u}{\arg\min} \left\{ \|D^u\, u_{k+1} + \rho^{-1}\,\alpha_k - z\|_2^2 \right\} \\ \alpha_{k+1} & = & \alpha_k + \rho\,(D^u\, u_{k+1} - z_{k+1}), \end{array}$$

where the update for z_{k+1} can be done using the algorithm from [47].

As explained in the main manuscript, in practice, using this update as part of an ADMM algorithm leads to difficulty enforcing the ℓ_1 constraint in reasonable runtimes, and results in larger reconstruction error than the technique we have recommended.

B.2 Solution path algorithm for finding the constrained updates

We now discuss the optimization procedure based on Theorem 3.5.1 to solve (3.6).

- First, using the solution path algorithm from [141], solve for every $\lambda > 0$,

$$\hat{\gamma}_\lambda = \underset{\|\gamma\|_\infty \leq \lambda}{\arg\min} \quad \frac{1}{2}\|\underline{Y} \times_2 v \times_3 w - (D^u)^T\gamma\|_2^2.$$

  This will produce a finite sequence of values $\lambda_1 \geq \lambda_2 \geq ....\geq \lambda_K$ that are the kinks of the coordinates of $\hat{\gamma}_\lambda$, which are piecewise linear functions (of λ).

- Next, construct the partition $\Gamma = \{[0, \lambda_K), [\lambda_K, \lambda_{K-1}), ..., [\lambda_2, \lambda_1]\}$ and for every interval $I \in \Gamma$, solve the following problem:

$$\underset{\lambda \in I}{\text{minimize}} \left[\|\underline{Y} \times_2 v \times_3 w - (D^u)^T\gamma_\lambda\|_2 + \lambda c_u\right].$$

  To do so, we exploit the fact that $\lambda \in I$ satisfies

$$\gamma_\lambda = \frac{\lambda}{\lambda_{i+1} - \lambda i}\left(\gamma_{\lambda_i+1} - \gamma_{\lambda_i}\right).$$

- Then, set

$$\lambda^* = \underset{0 \leq \lambda}{\arg\min} \left[\|\underline{Y} \times_2 v \times_3 w - (D^u)^T \hat{\gamma}_\lambda\|_2 + \lambda c_u \right].$$

Finally, the solution to (3.6) is

$$u^* = \frac{\left(\underline{Y} \times_2 v \times_3 w - (D^u)^T \hat{\gamma}_{\lambda^*}\right)}{\|\underline{Y} \times_2 v \times_3 w - (D^u)^T \hat{\gamma}_{\lambda^*}\|_2}.$$

B.3 Proof of technical results

B.3.1 Proof of Theorem 3.5.1

Throughout we define

$$x \ = \ \underline{Y} \times_2 v \times_3 w,$$

and $D = D^u$.

Next we note that the Lagrange dual function of the original problem is given by

$$\begin{aligned} L(\lambda, \mu) &= \underset{u}{\text{minimize}} \left[-x^T u + \lambda \left(\|Du\|_1 - c_S \right) + \mu \left(\|u\|_2^2 \ - \ 1 \right) \right] \\ &= \underset{u}{\text{minimize}} \left[-x^T u + \lambda \|Du\|_1 + \mu \|u\|_2^2 \right] - \mu - \lambda c_S \end{aligned}$$

subject to $\lambda, \mu \geq 0$.

Next, define for fixed $\lambda, \mu \geq 0$, the function $g_{\lambda,\mu}$: $\mathbb{R}^S \longrightarrow \mathbb{R}$ given by

$$g_{\lambda,\mu}(u) = -x^T u + \lambda \|Du\|_1 + \mu \|u\|_2^2. \tag{B.2}$$

From (D.2) we need to solve the following problem:

$$\underset{u}{\text{minimize}} \ g_{\lambda,\mu}(u) \ , \tag{B.3}$$

which can be rewriten as

$$\begin{aligned} &\underset{u,z}{\text{minimize}} \left[-x^T u + \lambda\|z\|_1 + \mu\|u\|_2^2\right] \\ &\text{subject to} \qquad z = Du\,. \end{aligned}$$

This problem has the following Lagrangian:

$$L_{\lambda,\mu}\left(z,u,\gamma\right) = -x^T u + \lambda\|z\|_1 + \mu\|u\|_2^2 + \gamma^T\left(Du - z\right)\,,$$

which is nicely separable in u and z.

Let us now consider some special cases of μ and λ. First, if $\lambda = 0$ and $\mu = 0$, then clearly,

$$\min_{z,u} L_{\lambda,\mu}\left(z,u,\gamma\right) = -\infty \qquad \forall\gamma.$$

Second, if $\lambda = 0$ and $\mu > 0$, then

$$\min_{u} \left[-x^T u + \lambda\|Du\|_1 + \mu\|u\|_2^2\right] = -\frac{1}{4\mu}x^T x.$$

Next, if $\lambda > 0$ and $\mu = 0$, then

$$\min_{z,u} L_{\lambda,\mu}\left(z,u,\gamma\right) = -\infty \qquad \forall\gamma \quad \text{with} \quad D^T\gamma \neq x,$$

and

$$\min_{z,u} L_{\lambda,\mu}\left(z,u,\gamma\right) = 0 \qquad \forall\gamma \text{ with} \quad D^T\gamma = x \quad \text{and} \quad \|\gamma\|_\infty \leq \lambda.$$

Thus

$$\min_{u} \left[-x^T u + \lambda\|Du\|_1\right] = \begin{cases} -\infty & \text{if } x \notin Range\left(D^T\right) \\ 0 & \text{if } \exists\gamma \text{ with} \quad D^T\gamma = x \quad \text{and} \quad \|\gamma\|_\infty \leq \lambda. \end{cases}$$

Finally, let us now focus on $\mu > 0$ or $\lambda > 0$. Then

$$\min_u \left[-x^T u + \mu \|u\|_2^2 + \gamma^T D u\right] = -\frac{1}{4\mu}\|x - D^T\gamma\|_2^2,$$

while (see [141])

$$\min_z \left[\lambda \|z\|_1 - \gamma^T z\right] = \begin{cases} 0 & \text{if} \quad \|\gamma\|_\infty \leq \lambda\,, \\ -\infty & \text{otherwise.} \end{cases}$$

Hence, the dual problem to (B.3) is equivalent to

$$\begin{aligned} \underset{\gamma}{\text{minimize}} \quad & \frac{1}{4\mu}\|x - D^T\gamma\|_2^2 \\ \text{subject to} \quad & \|\gamma\|_\infty \leq \lambda\,. \end{aligned}$$

But for $\mu > 0$ fixed, this is equivalent to solving the problem

$$\begin{aligned} &\underset{\gamma}{\text{minimize}}\, \frac{1}{2}\|x - D^T\gamma\|_2^2 \\ &s.t \qquad \|\gamma\|_\infty \leq \lambda\,, \end{aligned} \tag{B.4}$$

which can be solved for every $\lambda \geq 0$ using the solution path algorithm from [141].

Let us denote by $\hat{\gamma}_\lambda$ the solution to (B.4) for a fixed λ. Therefore,

$$L\left(\lambda, \mu\right) = -\frac{1}{4\mu}\|x - D^T\hat{\gamma}_\lambda\|_2^2 - \mu - \lambda c_S,$$

which implies that the dual to the original problem becomes

$$\underset{\lambda,\mu\geq 0}{\text{maximize}} \left[-\frac{1}{4\mu}\|x - D^T\hat{\gamma}_\lambda\|_2^2 - \mu - \lambda c_S\right]. \tag{B.5}$$

Finally, recall from [14] that any u^* solution to the original problem must also solve

$$u^* = \arg\min_u \left[-x^T u + \lambda^* \|Du\|_1 + \mu^* \|u\|_2^2\right],$$

for λ^* and μ^* that are optimal for (B.5). However, the objective function in (B.3) is strictly convex since $\mu^* > 0$, and so its solution u^* is unique and also solves

$$\begin{aligned} &\underset{u,z}{\text{minimize}} \left[-x^T u + \lambda^* \|z\|_1 + \mu^* \|u\|_2^2\right] \\ &\text{subejct to} \qquad z = Du. \end{aligned}$$

The KKT optimality conditions for this problem imply that

$$0 = \begin{pmatrix} -x + 2\mu^* u^* \\ \lambda^* \alpha \end{pmatrix} + \begin{pmatrix} D^T \gamma_{\lambda^*} \\ -\gamma_{\lambda^*} \end{pmatrix},$$

for some α subgradient of the function $z \to \|z\|_1$ at $z^* = D\,u^*$. Therefore

$$u^* = \frac{\left(x - D^T \hat{\gamma}_{\lambda^*}\right)}{\|x - D^T \hat{\gamma}_{\lambda^*}\|_2},$$

and the result follows.

The consequence for Corollary 3.5.2 comes from the fact that λ^* equals to the given λ in the unconstrained formulation of the problem. We also notice that, as in [141], $x - D^T \hat{\gamma}_{\lambda^*}$ is the solution to

$$\min_\beta \ \|x - \beta\|_2^2 \ + \ \lambda \|D\,\beta\|_1.$$

B.3.2 Proof of Theorem 3.6.1

Proof. Here we assume that data is generated as

$$\underline{Y} = d^* \, u^* \circ v^* \circ^* w + \epsilon$$

and

$$\|\hat{v} - v^*\|_2 < \frac{1}{\sqrt{2}}, \ \|\hat{w} - w^*\|_2 < \frac{1}{\sqrt{2}}.$$

Under these conditions we show that $\hat{u}$ defined as

$$\begin{aligned} \hat{u} = \quad & \underset{u \in \mathbb{R}^S}{\arg\min} \qquad - \underline{Y} \times_2 \hat{v} \times_3 \hat{w} \\ & \text{subject to} \quad \|u\|_2^2 \leq 1, \quad \|D^{(k_u+1)} u\|_1 \leq c_u \end{aligned}$$

satisfies

$$\begin{aligned} & \mathrm{P}\left(\tfrac{1}{2}\|u^* - \hat{u}\|_2^2 \leq \tfrac{1}{2} \frac{\frac{4\,t}{d^*} + \frac{c\, c_u \, L^{k_u + 1/2} \sqrt{\log L}}{d^*}}{\langle v^*, \hat{v} \rangle \, \langle w^*, \hat{w} \rangle - 2^{-1}} \right) \geq \\ & 1 - \sqrt{\tfrac{2}{\pi}} \frac{\exp\left(-t^2/2\right)}{t} - \frac{1}{L^{3/2}\sqrt{\log L}} \sqrt{\frac{2}{5\pi}}. \end{aligned}$$

for some constant $c > 0$. The proof will then follow by an application of this claim after each block update, and applying the identity for the intersection of such events.

To prove the claim above, we start by noticing that

$$\begin{aligned} \hat{u} = \quad & \underset{u \in \mathbb{R}^S}{\arg\min} \qquad - (d^*)^{-1} u^T \underline{Y} \times_2 \hat{v} \times_3 \hat{w} \\ & \text{subject to} \quad \|u\|_2^2 \leq 1, \quad \|D^{(k_u+1)} u\|_1 \leq c_u. \end{aligned}$$

Next we use the notation R for the row space of D: $R = row(D)$ and $R^{\perp} = null(D)$. Moreover, P_V denotes the perpendicular projection onto the space

V. Hence, by sub-optimality,

$$\begin{aligned}\tfrac{1}{2}\|\hat{u}-u^*\|_2^2 &\leq 1-\hat{u}^T u^* + \tfrac{1}{d^*}\left(\underline{Y}\times_2\hat{v}\times_3\hat{w}\right)^T(\hat{u}-u^*)\\ &= 1-\hat{u}^T u^* + \tfrac{1}{d^*}\left((d^*\,u^*\circ v^*\circ^* w+\epsilon)\times_2\hat{v}\times_3\hat{w}\right)^T\\ &\quad (P_R+P_{R^\perp})(\hat{u}-u^*)\\ &= 1-\hat{u}^T u^* + \langle\hat{v},v^*\rangle\langle\hat{w},w^*\rangle\,(u^*)^T(\hat{u}-u^*)+\\ &\quad \tfrac{1}{d^*}\epsilon\times_2\hat{v}\times_3\hat{w}\,(P_R+P_{R^\perp})(\hat{u}-u^*)\,.\end{aligned} \tag{B.6}$$

Let us now bound the terms in the expression above. First, let $a_1,\ldots,a_{k_u+1}$ be an orthonormal basis of $R^\perp$. Then

$$\begin{aligned}\tfrac{1}{d^*}\left(\epsilon\times_2\hat{v}\times_3\hat{w}\right)^T P_{R^\perp}(\hat{u}-u^*) &= \tfrac{1}{d^*}\sum_{j=1}^{k_u+1}\left(\left(\epsilon\times_2\hat{v}\times_3\hat{w}\right)^T a_j\right)(a_j^T(\hat{u}-u^*))\\ &\leq \tfrac{2}{d^*}\sum_{j=1}^{k_u+1}\left|\left(\epsilon\times_2\hat{v}\times_3\hat{w}\right)^T a_j\right|\\ &\leq \tfrac{2}{d^*}(\,(k_u+1)\,t)\end{aligned} \tag{B.7}$$

for some constant c with probability at least

$$1-(k_u+1)\sqrt{\frac{2}{\pi}}\frac{\exp\left(-t^2/2\right)}{t}.$$

Here we have used Mill's inequality.

Next we bound the term involving the projection operator onto the space R in (B.6). By Holder's inequality,

$$\begin{aligned}\tfrac{1}{d^*}\left(\epsilon\times_2\hat{v}\times_3\hat{w}\right)^T P_R\,(\hat{u}-u^*) \leq\ & \tfrac{1}{d^*}\left\|\left(\epsilon\times_2\hat{v}\times_3\hat{w}\right)^T\left(D^{(k_u+1)}\right)^-\right\|_\infty\cdot\\ & \left(\|D^{(k_u+1)}\hat{u}\|_1+\|D^{(k_u+1)}u^*\|_1\right),\end{aligned}$$

and hence, as in Corollary 4 from [149], we find that there exists a constant $c>0$ such that

$$\mathrm{P}\left(\frac{1}{d^*}\epsilon\times_2\hat{v}\times_3\hat{w}\,P_R\,(\hat{u}-u^*)\leq c\,\frac{L^{k_u+1/2}\sqrt{\log(L)}c_u}{d^*}\right)\geq 1-\frac{1}{L^{3/2}\sqrt{\log L}}\sqrt{\frac{2}{5\,\pi}}. \tag{B.8}$$

On the other hand, by the Cauchy–Schwarz inequality and the hypothesis, we have

$$\begin{aligned} 1 - \hat{u}^T u^* + \langle \hat{v}, v^* \rangle \langle \hat{w}, w^* \rangle (u^*)^T (\hat{u} - u^*) &= (1 - \langle \hat{v}, v^* \rangle \langle \hat{w}, w^* \rangle) \cdot \\ &\quad (\|u^*\|_2^2 - \langle \hat{u}, u^* \rangle) \\ &\leq (1 - \langle \hat{v}, v^* \rangle \langle \hat{w}, w^* \rangle) \|\hat{u} - u^*\|_2^2 \\ &\leq \tfrac{7}{16} \|\hat{u} - u^*\|_2^2. \end{aligned} \tag{B.9}$$

Combining (B.6), (B.7), (B.8), (B.9), and proceeding in similar fashion for the other updates, the identity

$$\mathrm{P}(A \cap B \cap C) = \mathrm{P}(A)\mathrm{P}(B \mid A)\mathrm{P}(C \mid A \cap B)$$

for any events A,B and C implies the result.

□

For the case of multiple factors, we have the following result. Suppose that the data is generated as

$$\underline{Y} \;=\; \sum_{j=1}^{J} d_j^* \, u_j^* \circ v_j^*, \circ, w_j^* \;+\; \bar{E}$$

where E is tensor of white noise. Suppose that we have current parameters estimates of $\{u_j^*\}_{j \neq j_0}$, $\{v_j^*\}_j$, $\{w_j^*\}_j$, $\{d_j^*\}_j$ which we denote by $\{\hat{u}_j\}_{j \neq j_0}$, $\{\hat{v}_j\}_j$, $\{\hat{w}_j\}_j$, $\{\hat{d}_j\}_j$.

Let us now provide an error bound for the estimate of $u_{j_0}^*$ given all

the other estimates. To that end, define

$$\hat{u}_{j_0} = \begin{array}{ll} \underset{u\in\mathbb{R}^L}{\arg\min} & \frac{1}{2}\left\| u - \left(\underline{Y} \times_2 \hat{v}_{j_0} \times_3 \hat{w}_{j_0} - \sum_{j\neq j_0} \hat{d}_j \, (\hat{v}_{j_0})^T \, \hat{v}_j \, (\hat{w}_{j_0})^T \, \hat{w}_j \, \hat{u}_j \right) \right\|_F^2 \\ \text{subject to} & \|D^u u\|_1 \, \leq \, c_u \\ & u^T u \, = \, 1, \end{array}$$

and assume that $\|D^u u^*_{j_0}\|_1 \leq c_u$ and

$$\|\hat{v}_{j_0} - v^*_{j_0}\|_2 < \frac{1}{\sqrt{2}}, \quad \|\hat{w}_{j_0} - w^*_{j_0}\|_2 < \frac{1}{\sqrt{2}}$$

This leads to the following lemma.

Lemma B.3.1. *Under the definitions just given,*

$$\begin{array}{l} P\left(\|u^*_{j_0} - \hat{u}_{j_0}\|_2^2 \leq 16 \left(\frac{4\,t}{d^*_{j_0}} + \frac{c\, c_u \, L^{k_u+1/2}\sqrt{\log L}}{d^*_{j_0}} \right) + 16\, U \right) \\ \geq 1 - \sqrt{\frac{2}{\pi}} \frac{\exp\left(-t^2/2\right)}{t} - \frac{1}{L^{3/2}\sqrt{\log L}} \sqrt{\frac{2}{5\pi}}, \end{array}$$

where

$$U = 2 \left\| \frac{1}{d^*_{j_0}} \sum_{j\neq j_0} \left(-\hat{d}_j \, \left(\hat{v}_j \cdot \hat{v}_{j_0}\right) \left(\hat{w}_j \cdot \hat{w}_{j_0}\right) \hat{u}_j + d^*_j \, \left(v^*_j \cdot \hat{v}_{j_0}\right) \left(w^*_j \cdot \hat{w}_{j_0}\right) \, u^*_j \right) \right\|_2 .$$

Proof. Follows immediately by sub-optimality. □

B.4 Simulation details

In our set of experiments we considered 5 different hidden rank-1 tensors constructed as $u \circ v \circ w$ where the vectors u, v and w are described below. The notation $\{x\}_i^j$ indicates that components i through j of the vector are all equal to the value x.

Structure 1

- $u = \{1, 1, 1, -1, -1, -1, 0, 0, 0, 0\}$.
- $v = \{0\}_1^{100}, \{1\}_{101}^{500}, \{0\}_{501}^{1000}$.
- $w = \{-1\}_1^{100}, \{0\}_{101}^{200}, \{1\}_{201}^{400}$.

Structure 2

- $u = \{0, 0, 0, -1, -1, -1, 0, 0, 0, 0\}$.
- $v = \{v_i\}_{i=1}^{1000}$ with $v_i = \cos\left(12\,\pi\,\frac{(i-1)}{999}\right)$ for $i = 1, 2, \ldots, 1000$.
- $w = \{w_i\}_{i=1}^{400}$ with $w_i = \cos\left(9\,\pi\,\frac{(i-1)}{399}\right)$ for $i = 1, 2, \ldots, 400$.

Structure 3

- $u = \{0, 0, 0, 0, -1, -1, 1, 1, 1, 1\}$.
- $v = \{v_i\}_{i=1}^{1000}$ with $v_i = \left(\frac{(i-1)}{999} - 0.7\right)^2 + \left(\frac{(i-1)}{999}\right)^2$ for $i = 1, 2, \ldots, 1000$.
- Define $w_i' = \frac{i-1}{399}$ for $i = 1, \ldots, 400$. Then, set $w_i = w_i'\,(0.05 - w_i')$ for $i = 1, \ldots, 200$ and $w_i = (w_i')^2$ for $i = 201, \ldots, 400$.

Structure 4

- $u = \{0, 0, 0, 0, 0, 1, 1, 1, 1, 1\}$

- Define $v_i' = \frac{i-1}{999}$ for $i = 1, \ldots, 1000$. Then,
  $v_i = cos(\pi\, v_i') + .65$.

- $w = \{0\}^{100}, \{1\}_{101}^{150}, \{0\}_{151}^{300}, \{1\}_{301}^{350}, \{0\}_{351}^{400}$.

Structure 5

- $u = \{-1, -1, 0, 0, 1, 1, 1, -1, -1, -1\}$.

- v has 80% of its entries equal to zero and the remanining 20% are random numbers drawn from a standar normal distribution.

- w has 92.5% of its entries equal to zero and the remanining 7.5% are random numbers drawn from a standar normal distribution.

B.5 Real data examples additional details

B.5.1 Flu hospitalizations

Our flu example uses aggregate, non-identifiable hospitalization records from each of the eight largest counties in Texas from January 1, 2003 to December 30, 2009. Our data-use agreement does not permit dissemination of these hospital records. We also use data on temperature and air quality (particulate matter) in these counties, which can be obtained directly from CDC Wonder (http://wonder.cdc.gov/).

B.5.2 Motion capture

To construct the tensors involved in the five task considered, we use the variables: the second coordinate for root (variable 2), the first coordinate for upperback (variable 10), the first coordinate for upperneck (variable 19), the first coordinate for head (variable 22), the second coordinate for rhumerus (variable 28), rradius (variable 30), the second coordinate for lhumerus (variable 40), lradius (variable 42), the second coordinate for lhand (variable 44), lfingers (variable 45), rtibia (variable 52), ltibia (variable 59).

For task 138–story we use videos corresponding to subject 138 in the moCap repository. Videos 11-14 are used to construct the training tensor while 15-18 are used to build the test tensor.

To build task 107 walking we use videos from subject 107. For training we use videos 1-4 for training while videos 5-8 are used for testing.

For task 09-run we use videos corresponding to subject 9. Videos 1-4 are used for training, and videos 5-8 are used for testing.

To construct task 138 marching we take videos from subject 138. For training we use videos 1-4 for training while videos 5-8 are used for testing.

Finally, for task 126, the training set is built using videos 1,3,6,8 while the test set uses videos 2,4,7,9.

Appendix C

Proofs of theorems for Chapter 4

C.1 Proof of Theorem 4.3.1

Proof. Let us assume that $\hat{\theta}$ solves (4.3). Then, we define $\hat{g}_i = \hat{\theta}_i - \log(n\,\delta_n)$ and $c = \|\Delta^{(k+1)}\hat{\theta}\|_q^p$. Hence from the KKT conditions (4.3) is equivalent to

$$\begin{array}{ll} \underset{\theta}{\text{minimize}} & \sum_{i=1}^{D_n} \{\exp(\theta_i) - x_i\theta_i\} \\ \text{subject to} & \|\Delta^{(k+1)}\theta\|_q^p \leq c. \end{array}$$

Now, with the change of variable $\theta = g + \log(n\,\delta_n)$ and dividing by n this is equivalent to

$$\begin{array}{ll} \underset{g}{\text{minimize}} & \delta_n \sum_{i=1}^{D_n} \exp(g_i) - \frac{1}{n}\sum_{i=1}^{D_n} x_i\, g_i \\ \text{subject to} & \|\Delta^{(k+1)}g\|_q^p \leq c. \end{array} \tag{C.1}$$

Next we define the function

$$G(g) = \delta_n \sum_{i=1}^{D_n} \exp(g_i) - \frac{1}{n}\sum_{i=1}^{D_n} x_i\, g_i.$$

and for an arbitrary $g \in \mathbb{R}^{D_n}$ we define $g' \in \mathbb{R}^{D_n}$ as

$$g_i' = g_i - \log\left(\delta_n \sum_{j=1}^{D_n} \exp(g_j)\right).$$

Then

$$G(g') = G(g) + 1 - \delta_n \sum_{j=1}^{D_n} \exp(g_j) + \log\left(\delta_n \sum_{j=1}^{D_n} \exp(g_j)\right) \leq G(g),$$

since $t - \log(t) \geq 1$ for all $t > 0$. Moreover,

$$\|\Delta^{(k+1)} g\|_q^p = \|\Delta^{(k+1)} g'\|_q^p.$$

Therefore, problem (C.1) is equivalent to

$$\begin{array}{ll} \underset{g}{\text{minimize}} & -\frac{1}{n}\sum_{i=1}^{D_n} x_i\, g_i \\ \text{subject to} & \delta_n \sum_{i=1}^{D_n} \exp(g_i) = 1 \\ & \|\Delta^{(k+1)} g\|_q^p \leq c, \end{array}$$

and the claim follows.

□

C.1.1 Proof of Theorem 4.4.1

Before beginning the proof of the claim we start by proving an auxiliary lemma.

Lemma C.1.1. *With the notation from Theorem 4.4.1, if $a \in \mathbb{R}^{D_n}$, then*

$$P\left(\left|\left(x - \exp(\theta^0)\right)^T a\right| \geq C_r \frac{n\,\|a\|_\infty}{D_n^r}\right) \leq 4\exp\left(-c_r \frac{n}{D_n^{2r}}\right)$$

for all $r > 0$ and some positive constants C_r and c_r depending on r.

Proof. Our proof is inspired by the construction in Lemma 3 from [41]. We start by denoting $p_i = \frac{\exp\left(\theta_i^0\right)}{n}$, $i = 1, \ldots, D_n$. Then we can think of x_i as the occurrences of value i among $u_1, \ldots, u_n$ where $\mathrm{P}(u_k = j) = p_j$ for $j = 1, ...D_n$ and $k = 1, 2, \ldots$. Next, we define $N \sim \text{Poisson}\,(n)$, and x_i' as the occurrences of value i among $u_1, \ldots, u_N$. Clearly, $x_i' \sim \text{Poisson}\,(n\,p_i)$. Moreover,

$$\left|\sum_{i=1}^{D_n} a_i\,(x_i - n\,p_i)\right| \leq \left|\sum_{i=1}^{D_n} a_i\,(x_i' - n\,p_i)\right| + \left|\sum_{i=1}^{D_n} a_i\,(x_i - x_i')\right|,$$

form which

$$\begin{array}{rcl} \mathrm{P}\left(\left|\sum_{i=1}^{D_n} a_i\,(x_i - n\,p_i)\right| \geq 2\epsilon\right) & \leq & \mathrm{P}\left(\|a\|_\infty |N-n| \geq \epsilon\right) \; + \\ & & \mathrm{P}\left(\left|\sum_{i=1}^{D_n} a_i\,(x'_i - n\,p_i)\right| \geq \epsilon\right) \end{array} \tag{C.2}$$

for all $\epsilon > 0$. We now bound both terms in (C.2). First, we proceed using Hoeffding's inequality,

$$\begin{array}{rcl} \mathrm{P}\left(\sum_{i=1}^{D_n} a_i\,(x'_i - n\,p_i) \geq \epsilon\right) & \leq & \inf\limits_{t>0} \exp\left(-\epsilon\,t + \sum_{i=1}^{D_n} n\,p_i\,(\exp(t\,a_i) - 1 - t\,a_i)\right) \\ & \leq & \inf\limits_{t>0} \exp\left(-\epsilon\,t + n\,\left(\exp(t\,\|a\|_\infty) - 1 - t\,\|a\|_\infty\right)\right) \\ & \leq & \exp\left(-\frac{\epsilon}{D_n^r\,\|a\|_\infty} + n\left(\exp(\frac{1}{D_n^r}) - 1 - \frac{1}{D_n^r}\right)\right) \\ & \leq & \exp\left(-\frac{\epsilon}{D_n^r\,\|a\|_\infty} + \frac{c\,n}{D_n^{2r}}\right) \end{array}$$

for some positive constant c if D_n^r is large enough. Therefore, setting $\epsilon = c_1\,n\,\|a\|_\infty\,D_n^{-r}$ with $c_1 > c$, we obtain

$$\mathrm{P}\left(\sum_{i=1}^{D_n} a_i\,(x'_i - n\,p_i) \geq c_1 \frac{n\,\|a\|_\infty}{D_n^r}\right) \leq \exp\left(-\frac{(c_1 - c)n}{D_n^{2r}}\right).$$

With union bound inequality and repeating the same argument from above, we arrive to

$$\mathrm{P}\left(|\sum_{i=1}^{D_n} a_i\,(x'_i - n\,p_i)| \geq c_1 \frac{n\,\|a\|_\infty}{D_n^r}\right) \leq 2\exp\left(-\frac{(c_1 - c)n}{D_n^{2r}}\right).$$

Finally, from the proof of Lemma 3 in [41] we have

$$\mathrm{P}\left(\|a\|_\infty |N-n| \geq c_1 \frac{n\,\|a\|_\infty}{D_n^r}\right) \leq 2\,\exp\left(-\frac{c_1^2}{4}\frac{n}{D_n^{2r}}\right)$$

and the result follows.

□

Proof. Let e_1 an element of the canonical basis in $\mathbb{R}^{D_n}$ and let us denote by P the orthogonal projection onto R, the row space of Δ^{k+1}. We start by noticing that from sub-optimality we have

$$l(\hat{\theta}) + \tau \left\| \Delta^{(k+1)} \hat{\theta} \right\|_1 \leq l(\theta^0) + \tau \left\| \Delta^{(k+1)} \theta^0 \right\|_1.$$

Hence, we obtain

$$\begin{aligned} \sum_{j=1}^{D_n} \delta_n \, f_0(\xi_j') \log \left(\frac{f_0(\xi_j')}{\hat{f}(\xi_j')} \right) \quad &\leq \quad \tfrac{1}{n} \left(x - \exp(\theta^0) \right)^T \left(\left(\Delta^{(k+1)} \right)^- \Delta^{(k+1)} + P_{R^\perp} \right) \cdot \\ &\left(\hat{\theta} - \theta^0 \right) \; + \; \tfrac{\tau}{n} \left(\left\| \Delta^{(k+1)} \theta^0 \right\|_1 - \left\| \Delta^{(k+1)} \hat{\theta} \right\|_1 \right). \end{aligned} \tag{C.3}$$

Next we bound each of the terms on the right hand side of (C.3). First, define $v_1, \ldots, v_{k+1}$ to be an orthonormal basis of $R^\perp$ such that $v_1 = D_n^{-1/2} (1, \ldots, 1)$. Then, it is not difficult to see that these vectors can be chosen to satisfy $\|v_j\|_\infty = O(D_n^{-1/2})$ for $j = 1, \ldots, k+1$. Therefore, by Holder's inequality

$$\begin{aligned} \tfrac{1}{n} \left(x - \exp(\theta^0) \right)^T \, P_{R^\perp} \left(\hat{\theta} - \theta^0 \right) \quad &= \quad \tfrac{1}{n} \sum_{j=1}^{k+1} \left[\left(x - \exp(\theta^0) \right)^T \, v_j \right] \left[v_j^T \left(\hat{\theta} - \theta^0 \right) \right] \\ &\leq \quad \tfrac{1}{n} \sum_{j=1}^{k+1} \left[\left(x - \exp(\theta^0) \right)^T \, v_j \right] D_n^{1/2} \\ &\quad \cdot \left(\| \log(f_0(\xi')) \|_\infty + \| \log(\hat{f}(\xi')) \|_\infty \right). \end{aligned} \tag{C.4}$$

It follows form the previous lemma that

$$\frac{1}{n} \left(x - \exp(\theta^0) \right)^T \, P_{R^\perp} \left(\hat{\theta} - \theta^0 \right) = O_{\mathrm{P}} \left(\frac{1}{n^{r/s-b}} \right).$$

assuming that we constraint $\|\hat{\theta} - \log(n \, \delta_n)\|_\infty \leq n^b$.

On the other hand,

$$\begin{aligned} &\tfrac{1}{n} \left(x - \exp(\theta^0) \right)^T \left(\left(\Delta^{(k+1)} \right)^- \Delta^{(k+1)} \right) \left(\hat{\theta} - \theta^0 \right) \leq \\ &\tfrac{1}{n} \left\| \left(x - \exp(\theta^0) \right)^T \left(\Delta^{(k+1)} \right)^- \right\|_\infty \left(\left\| \Delta^{(k+1)} \theta^0 \right\|_1 + \left\| \Delta^{(k+1)} \hat{\theta} \right\|_1 \right). \end{aligned} \tag{C.5}$$

Moreover, from the previous lemma we obtain

$$\mathrm{P}\left(\left\|(x-\exp(\theta^0))^T\left(\Delta^{(k+1)}\right)^-\right\|_\infty \geq C_r\,\frac{n\,\|\left(\Delta^{(k+1)}\right)^-\|_\infty}{D^r}\right) \leq$$
$$4\exp\left(-c_r\,\frac{n}{D_n^{2r}}+\log(D_n)\right)$$

Therefore, combining $(C.3)$, $(C.4)$ and $(C.5)$, if

$$\tau \geq \|\left(x-\exp(\theta^0)\right)^T\left(\Delta^{(k+1)}\right)^-\|_\infty,$$

then,

$$\sum_{j=1}^{D}\delta\,f(z_j)\log\left(\frac{f(z_j)}{\hat{f}(z_j)}\right) \;\leq\; O_{\mathrm{P}}\left(\frac{\|\left(\Delta^{(k+1)}\right)^-\|_\infty}{D_n^r}\|\Delta^{(k+1)}\theta^0\|_1+\frac{n^b}{D_n^r}\right) \qquad \text{(C.6)}$$

□

Appendix D

Proofs and experiments details for Chapter 5

D.1 Gradient expression for ℓ_2 regularization

Here we write the mathematical expressions for the gradient of the objective function when performing L2 deconvolution, As in Section 3.3 of the main document. Using the notation there, we have that

$$[\nabla l(\theta)]_j = \sum_{i=1}^{D} G_{ij} e^{\theta_j} \left(\frac{x_i}{\lambda_i(\theta)} - 1 \right) ,$$

and

$$\nabla \|\Delta^{(k+1)}\theta\|_2^2 = 2 \left(\Delta^{(k+1)}\right)^T \Delta^{(k+1)}\theta \, .$$

D.1.1 Proof of Theorem 5.5.1

Proof. Motivated by [62], given $\alpha \in A$ we define the function $F(\xi, \alpha) = (\phi * \alpha)(\xi)$ for $\mu \in \mathbb{R}$. Clearly, $F(\xi, \alpha)$ is a density that induces a measure in $\mathbb{R}$ that is absolutely continuous with respect to the Lebesgue measure in $\mathbb{R}$. Also, we observe that if $\alpha, \beta \in A$, then, for any Borel measurable set E, we have by Tonelli's theorem that

$$\begin{array}{rcl} \left| \int_E \phi * \alpha(\mu) d\mu - \int_E \phi * \beta(\mu) d\mu \right| & = & \left| \int_{\mathbb{R}} \left(\int_E \phi(\mu - y) d\mu \right) (\alpha(y) - \beta(y)) \, dy \right| \\ & \leq & d(\alpha, \beta). \end{array}$$

Hence $d(\alpha, \beta) = 0$ implies that $\phi * \alpha$ and $\phi * \beta$ induce the same probability measures in $(\mathcal{R}, \mathcal{B}(\mathcal{R}))$.

Next we verify the assumptions in Theorem 1 from [62]. This is done into different steps below. Steps 1-4 verify the assumptions B1-B4 in Theorem 1 from [62]. Steps 5-6 are needed in the general case in which the data is binned. These are also related to ideas from [148].

Step 1

Given $\alpha \in A$ and $\epsilon > 0$, the function

$$\xi \to \sup_{\beta \in S_m : d(\alpha,\beta) < \epsilon} (\phi * \beta)(\xi)$$

is continuous and therefore measurable on ξ. To see this, simply note that for any $\beta \in A$ we have that

$$\| (\phi * \beta)' \|_\infty = \| (\phi' * \beta) \|_\infty \leq \|\phi'\|_\infty \int_{\mathcal{R}} \beta(\mu) d\mu = \|\phi'\|_\infty.$$

Hence all the functions $\beta \in S_m$ are $(\|\phi'\|_\infty + 1)$-Lipschitz and the claim follows. Also, we note that

$$\lim_{\epsilon \to 0} \sup_{\beta \in S_m : d(\alpha,\beta) < \epsilon} (\phi * \beta)(\xi) = \phi * \alpha(\xi).$$

This follows by noticing that

$$\begin{aligned} \left| \sup_{\beta \in S_m : d(\alpha,\beta) < \epsilon} (\phi * \beta)(\xi) - \phi * \alpha(\xi) \right| &\leq \sup_{\beta \in S_m : d(\alpha,\beta) < \epsilon} |(\phi * (\beta - \alpha))(\xi)| \\ &\leq \|\phi\|_\infty \sup_{\beta : d(\alpha,\beta) < \epsilon} d(\alpha, \beta) \\ &\leq \epsilon \|\phi\|_\infty. \end{aligned}$$

Step 2

Define $E_\alpha(g) := \int_{\mathbb{R}} g(\xi)(\phi * \alpha)(\xi) d\xi$ for any function g. Then for any $\alpha \in A$ and $\epsilon > 0$ we have

$$\begin{aligned} E_{f_0}\left(\log\left(\sup_{\beta \in S_m : d(\alpha,\beta)<\epsilon} (\phi * \beta)(\xi)\right)\right) &\leq E_{f_0}\left(\log\left(\sup_{\beta : d(\alpha,\beta)<\epsilon} (\phi * \beta)(\xi)\right)\right) \\ &\leq \int_{\mathcal{R}} \log\left(\|\phi\|_\infty\right) \phi * f_0(\xi)\, d\xi \\ &< \infty. \end{aligned}$$

Step 3

Next we show that S_m is compact on (A, d). Throughout, we use the notation $\to_u$ to indicate uniform convergence. To show the claim, choose $\{\alpha_l\}$ a sequence in S_m. Then since $\{(\log(\alpha_l))^{(k+1)}\}$ are T_m-Lipschitz and uniformly bounded it follows by Arzela-Ascoli Theorem that there exists a sub-sequence $\{\alpha_{1,l}\} \subset \{\alpha_l\}$ such that $(\log(\alpha_{1,l}))^{(k+1)} \to_u g_{k+1}$ in $[-1, 1]$ for some function $g_{k+1} : [-1, 1] \to \mathcal{R}$ which is also T_m-Lipschitz. Note that we can again use Arzela-Ascoli Theorem applied to the sequence $\{\alpha_{1,l}\}$ to ensure that there exists a sub-sequence $\{\alpha_{2,l}\} \subset \{\alpha_{1,l}\}$ such that $(\log(\alpha_{1,l}))^{(k+1)} \to_u g_{k+1}$, in $[-2, 2]$. Thus we extend the domain of g_{k+1} if necessary.

Proceeding by induction we conclude that for every $N \in \mathbb{N}$ there exists a sequence $\{\alpha_{N,l}\}_{l\in\mathbb{N}} \subset \{\alpha_{N-1,l}\}_{l\in\mathbb{N}}$ such that

$$(\log(\alpha_{N,l}))^{(k+1)} \to_u g_{k+1}, \quad \text{as} \quad l \to \infty,$$

in $[-N, N]$ as $l \to \infty$. Hence with Cantor's diagonal argument we conclude that there exists a sub-sequence $\{\alpha_{l_j}\} \subset \{\alpha_l\}$ such that

$$\left(\log(\alpha_{l_j})\right)^{(k+1)} \to_u g_{k+1} \quad \text{as} \quad j \to \infty,$$

in $[-N, N]$ for all $N \in \mathbb{N}$. Since $|\left(\log(\alpha_{l_j})\right)^{(k)}(0)| \leq T_m$ for all j. Then without loss of generality, we can assume that

$$\left(\log(\alpha_{l_j})\right)^{(k)} \to_u g_k \text{ as } j \to \infty,$$

in $[-N, N]$ for all $N \in \mathbb{N}$ and where the function g_k satisfies $g_k' = g_{k+1}$. Continuing with this process we can assume, without loss of generality, that

$$\log(\alpha_{l_j}) \to_u g \text{ as } j \to \infty,$$

in $[-N, N]$ for all $N \in \mathbb{N}$ for some function g satisfying $g^{(j)} = g_j$ for all $0 \leq j \leq k+1$. Therefore,

$$\alpha_{l_j} \to_u \exp(g) \text{ as } j \to \infty, \tag{D.1}$$

in $[-N, N]$ for all $N \in \mathbb{N}$.

Let us now prove that $\exp(g) \in S_m$. First, we observe by the Fatou's lemma e^g is integrable in $\mathcal{R}$ with respect to the Lebesgue measure. since S_m is tight and, we obtain

$$d\left(\exp(g), \alpha_{l_j}\right) \to 0.$$

This clearly also implies that $\exp(g)$ integrates to 1 or $\exp(g) \in \mathcal{P}$. Note that also by Fatou's lemma we have that $J_{k,q}(g) \leq K_m$ and by construction,

$$\max\left(\|\exp(g)\|_\infty, \|g^{(k+1)}\|_\infty, |g^{(k)}(0)|, \ldots, |g(0)|\right) \leq T_m.$$

Finally, combining all of this with g^{k+1} being T_m-Lipschitz, we arrive to $\exp(g) \in S_m$.

Step 4 By assumption (4), we have that

$$\sup_{\alpha \in A_m} d\left(f_0, \alpha\right) \to 0, \quad \text{as} \quad m \to \infty.$$

Step 5

Let us show that

$$\lim_{\epsilon \to 0} E_{f_0}\left(\sup_{d(\alpha,\beta)<\epsilon, \beta \in S_m} \log\left(\phi * \beta\right)\right) = E_{f_0}\left(\log\left(\phi * \alpha\right)\right)$$

for all $\alpha \in S_m$. First, note that for all ξ

$$0 \leq \max\left\{0, \sup_{d(\alpha,\beta)<\epsilon, \beta \in S_m} \log\left(\phi * \beta\right)(\xi)\right\} \leq \max\left\{0, \log\left(\|\phi\|_\infty\right)\right\}.$$

Hence, by Step 1 we obtain

$$\begin{aligned} 0 \leq \lim_{\epsilon \to 0} E_{f_0}\left(\max\left\{0, \sup_{d(\alpha,\beta)<\epsilon, \beta \in S_m} \log\left(\phi * \beta\right)\right\}\right) &= E_{f_0}\left(\max\left\{\log\left(\phi * \alpha\right), 0\right\}\right) \\ &< \infty. \end{aligned}$$

Now we observe that

$$0 \leq -\min\left\{0, \sup_{d(\alpha,\beta)<\epsilon, \beta \in S_m} \log\left(\phi * \beta\right)(\xi)\right\} \leq -\min\left\{0, \log\left(\phi * \alpha(\xi)\right)\right\},$$

and the claim follows from the monotone convergence theorem.

If $x_j = 1$ and $\xi_j = y_j$ for all $j = 1, \ldots, D_n$, the claim of Theorem 5.5.1 follows from Theorem 1 from [62]. Otherwise, we continue the proof below. In either case we can see that the solution set M_m^n is not empty given that the map $\alpha \to \phi * \alpha(\xi)$ is continuous with respect to the metric d for any ξ.

Step 6

Note that, by Glivenko-Cantelli Theorem and our assumption on the maximum number of bins, we have that, almost surely, the random distribution

$$G_n(\xi) = \sum_{j=1}^{D_n} \frac{x_j}{n} I_{(-\infty,\xi])}(\xi_j)$$

converges weakly to the distribution function associated with $\phi * f_0$. Hence, almost surely, from the Portmanteau theorem we have for any $\alpha \in S_{m_n}$ and any $\delta > 0$ it holds that

$$\limsup_{l\to\infty} \sum_{j=1}^{D_l} \frac{x_j}{l} \sup_{d(\alpha,\beta)<\delta,\beta\in S_m} \log\left(\phi * \beta\right)(\xi_j) \;\leq\; E_{f_0}\left(\sup_{d(\alpha,\beta)<\delta,\beta\in S_m} \log\left(\phi * \beta\right)(\xi)\right), \tag{D.2}$$

since the function

$$\xi \;\to\; \sup_{d(\alpha,\beta)<\delta,\beta\in S_m} \log\left(\phi * \beta\right)(\xi),$$

is continuous and bounded by above.

Next we define

$$m_1 = \min\left\{m : \sup_{\alpha\in A_m} d\left(\alpha, f_0\right) < \frac{1}{2}\right\}.$$

Clearly, $\beta_1, \beta_2 \in A_{m_1}$ implies $d(\beta_1, \beta_2) < 1$. Also, we see that the set $\Pi_1 := \{\alpha \in S_{m_1} : d\left(\alpha, f_0\right) \geq 1\} \subset S_{m_1} - A_{m_1}$ is d-compact. Hence, there exists $\alpha_1^1, \ldots, \alpha_{h_1}^1$ in Π_1 such that $\Pi_1 \subset \cup_{l=1}^{h_1}\{\alpha \in \Pi_1 : d\left(\alpha, \alpha_l^1\right) < \delta_{1,l}\}$ for positive constants $\{\delta_{1,l}\}$ satisfying that

$$E_{f_0}\left(\sup_{d(\alpha,\alpha_l^1)<\delta_{1,l},\alpha\in\Pi_1} \log\left(\phi * \alpha\right)\right) < E_{f_0}\left(\log\left(\phi * f_{m_1}\right)\right)$$

for $l = 1, \ldots, h_1$. Therefore from our assumptions on the sets A_m and also from (D.2), we arrive at

$$\begin{aligned}
&\limsup_{r\to\infty} \sum_{j=1}^{D_r} \frac{x_j}{r} \left(\sup_{d(\alpha,\alpha_l^1)<\delta_{1,l},\alpha\in\Pi_1} \log\left(\phi * \alpha\right)(\xi_j) - \log\left(\phi * f_{m_1}(\xi_j)\right) \right) \\
&\leq E_{f_0} \left(\sup_{d(\alpha,\alpha_l)<\delta_l,\alpha\in\Pi_1} \log\left(\phi * \alpha\right) \right) - E_{f_0}\left(\log\left(\phi * f_{m_1}\right)\right) \\
&< 0, \quad \text{a.s.}
\end{aligned}$$

Hence

$$\limsup_{r\to\infty} \sum_{j=1}^{D_r} x_j \left(\sup_{d(\alpha,\alpha_l^1)<\delta_{1,l},\alpha\in\Pi_1} \log\left(\phi * \alpha\right)(\xi_j) - \log\left(\phi * f_{m_1}(\xi_j)\right) \right) = -\infty \text{ a.s.}$$

This implies

$$\lim_{r\to\infty} \frac{\sup_{\alpha\in\Pi_1} \prod_{j=1}^{D_r} \phi * \alpha(\xi_j)^{x_j}}{\prod_{j=1}^{D_r} \phi * f_{m_1}(\xi_j)^{x_j}} = 0 \text{ a.s.}$$

Next we define

$$k_1 = \min \left\{ k_0 : r \geq k_0 \text{ implies } \frac{\sup_{\alpha\in\Pi_1} \prod_{j=1}^{D_r} \phi * \alpha(\xi_j)^{x_j}}{\prod_{j=1}^{D_r} \phi * f_{m_1}(\xi_j)^{x_j}} < 1 \right\}$$

and we set $m_{k_1} = m_1$. Therefore,

$$\sup_{\alpha\in M_{m_{k_1}}^{k_1}} d\left(\alpha, f_0\right) < 1.$$

Next we define m_2 as

$$m_2 = \min \left\{ m \geq m_1 : \sup_{\alpha\in A_m} d\left(\alpha, f_0\right) < \frac{1}{4} \right\} + 1$$

Then, $\beta_1, \beta_2 \in A_{m_2}$ implies $d(\beta_1, \beta_2) < 1/2$. We also see that $\Pi_2 := \{\alpha \in S_{m_2} : d\left(\alpha, f_0\right) \geq 1/2\} \subset S_{m_2} - A_{m_2}$ is d-compact. Hence there exists $\alpha_1^2, \ldots, \alpha_{h_2}^2$

in Π_2 such that $\Pi_2 \subset \cup_{l=1}^{h_2}\{\alpha : d(\alpha, \alpha_l^2) < \delta_{2,l}\}$ for positive constants $\{\delta_{2,l}\}$ satisfying

$$E_{f_0}\left(\sup_{d(\alpha,\alpha_l^2)<\delta_{2,l},\alpha\in\Pi_2} \log(\phi * \alpha)\right) < E_{f_0}(\log(\phi * f_{m_2}))$$

for $l = 1, \ldots, h_2$. Therefore,

$$\begin{aligned}
&\limsup_{r\to\infty} \sum_{j=1}^{D_r} \frac{x_j}{r}\left(\sup_{d(\alpha,\alpha_l^2)<\delta_{2,l},\alpha\in\Pi_2} \log(\phi * \alpha)(\xi_j) - \log(\phi * f_{m_2}(\xi_j))\right) \\
&\leq E_{f_0}\left(\sup_{d(\alpha,\alpha_l^2)<\delta_{2,l},\alpha\in\Pi_2} \log(\phi * \alpha)\right) - E_{f_0}(\log(\phi * f_{m_2})) \\
&< 0 \text{ a.s.}
\end{aligned}$$

So proceeding as before we obtain

$$\lim_{r\to\infty} \frac{\sup_{\alpha\in\Pi_2} \prod_{j=1}^{D_r} \phi * \alpha(\xi_j)^{x_j}}{\prod_{j=1}^{D_r} \phi * f_{m_2}(\xi_j)^{x_j}} = 0 \text{ a.s}$$

Finally we define

$$k_2 = \min\left\{k_0 : k_0 \geq k_1 \text{ and } r \geq k_0 \text{ implies } \frac{\sup_{\alpha\in\Pi_2} \prod_{j=1}^{D_r} \phi * \alpha(\xi_j)^{x_j}}{\prod_{j=1}^{D_r} \phi * f_{m_2}(\xi_j)^{x_j}} < 1\right\}$$

and we set $m_k = m_1$ for all $k_1 \leq k < k_2$ and $m_{k_2} = m_2$. By construction, we have that

$$\sup_{\alpha\in M_{m_{k_2}}^{k_2}} d(\alpha, f_0) < 1/2.$$

Thus an induction argument allow us to conclude the proof.

□

Bibliography

[1] Carnegie mellon university. graphics lab motion capture database.

[2] Ittai Abraham and Ofer Neiman. Using petal-decompositions to build a low stretch spanning tree. *ACM Symposium on Theory of Computing*, 44:395–406, 2012.

[3] Genevera Allen. "Sparse higher-order principal components analysis". In *International Conference on Artificial Intelligence and Statistics*, pages 27–36, 2012.

[4] Animashree Anandkumar, Rong Ge, and Majid Janzamin. "Guaranteed Non-Orthogonal Tensor Decomposition via Alternating Rank-1 Updates". *arXiv preprint arXiv:1402.5180*, 2014.

[5] Taylor Arnold, Veeranjaneyulu Sadhanala, and Ryan J. Tibshirani. *glmgen: Fast generalized lasso solver*. https://github.com/statsmaths/glmgen, 2014. R package version 0.0.2.

[6] Taylor Arnold and Ryan Tibshirani. "Efficient Implementations of the Generalized Lasso Dual Path Algorithm". *Journal of Computational and Graphical Statistics*, (just-accepted), 2015.

[7] Taylor B. Arnold and Ryan J. Tibshirani. *genlasso: Path algorithm for generalized lasso problems*, 2014. R package version 1.3.

[8] Álvaro Barbero and Suvrit Sra. Modular proximal optimization for multidimensional total-variation regularization. *arXiv preprint arXiv:1411.0589*, 2014.

[9] Andrew R Barron and Chyong-Hwa Sheu. Approximation of density functions by sequences of exponential families. *The Annals of Statistics*, pages 1347–1369, 1991.

[10] Mikhail Belkin and Partha Niyogi. Using manifold structure for partially labelled classification. *Advances in Neural Information Processing Systems*, 15, 2002.

[11] Richard Bellman. On the approximation of curves by line segments using dynamic programming. *Communications of the ACM*, 4(6):284, 1961.

[12] Aditya Bhaskara, Moses Charikar, Ankur Moitra, and Aravindan Vijayaraghavan. "Smoothed analysis of tensor decompositions". pages 594–603, 2014.

[13] Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato, and Jonathan Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends in Machine Learning*, 3(1):1–122, 2011.

[14] Stephen Boyd and Lieven Vandenberghe. "Convex optimization ". *Cambridge Univ. Pr*, 2004.

[15] Yuri Boykov, Olga Veksler, and Ramin Zabih. Fast approximate energy minimization via graph cuts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(11):1–18, 2001.

[16] Leo Breiman, Jerome Friedman, Charles J Stone, and Richard A Olshen. *Classification and regression trees*. CRC press, 1984.

[17] Lawrence Brown, Tony Cai, Ren Zhang, Linda Zhao, and Harrison Zhou. The root–unroot algorithm for density estimation as implemented via wavelet block thresholding. *Probability theory and related fields*, 146(3-4):401–433, 2010.

[18] Lawrence D Brown and Eitan Greenshtein. Nonparametric empirical bayes and compound decision approaches to estimation of a high-dimensional vector of normal means. *The Annals of Statistics*, pages 1685–1704, 2009.

[19] Tony Cai. Adaptive wavelet estimation: a block thresholding and oracle inequality approach. *Annals of statistics*, pages 898–924, 1999.

[20] Raymond Carroll, Aurore Delaigle, and Peter Hall. Deconvolution when classifying noisy data involving transformations. *Journal of the American Statistical Association*, 107(499):1166–1177, 2012.

[21] Raymond J Carroll and Peter Hall. Optimal rates of convergence for deconvolving a density. *Journal of the American Statistical Association*, 83(404):1184–1186, 1988.

[22] Gilles Celeux, Florence Forbes, Christian P. Robert, and D. Michael Titterington. Deviance information criteria for missing data models. *Bayesian analysis*, 1(4):651–673, 2006.

[23] Antonin Chambolle and Jérôme Darbon. On total variation minimization and surface evolution using parametric maximum flows. *International Journal of Computer Vision*, 84(3):288–307, 2009.

[24] Antonin Chambolle and Thomas Pock. A first-order primal-dual algorithm for convex problems with applications to imaging. *Journal of Mathematical Imaging and Vision*, 40:120–145, 2011.

[25] Andrzej Cichocki. "Tensor Decompositions: A New Concept in Brain Data Analysis?". *Journal of Control Measurement, and System Integration*, 7:507–517, 2011.

[26] Andrzej Cichocki, Rafal Zdunek, Anh Huy Phan, and Shun-ichi Amari. *"Nonnegative matrix and tensor factorizations: applications to exploratory multi-way data analysis and blind source separation"*. John Wiley & Sons, 2009.

[27] Ronald Coifman and Mauro Maggioni. Diffusion wavelets. *Applied and Computational Harmonic Analysis*, 21(2):53–94, 2006.

[28] Laurent Condat. A direct algorithm for 1d total variation denoising. *HAL preprint hal-00675043*, 2012.

[29] Thomas Cormen, Clifford Stein, Ronald Rivest, and Charles Leiserson. *Introduction to Algorithms*. McGraw-Hill Higher Education, 2nd edition, 2001.

[30] Mark Crovella and Eric Kolaczyk. Graph wavelets for spatial traffic analysis. *Annual Joint Conference of the IEEE Computer and Communications IEEE Societies*, 3:1848–1857, 2003.

[31] Gabor Csardi and Tamas Nepusz. The igraph software package for complex network research. *InterJournal*, Complex Systems:1695, 2006.

[32] Arnak Dalalyan, Mohamed Hebiri, and Johannes Lederer. On the prediction performance of the lasso. *To appear, Bernoulli*, 2014.

[33] P. Laurie Davies and Arne Kovac. Local extremes, runs, strings and multiresolution. *Annals of Statistics*, 29(1):1–65, 2001.

[34] P. Laurie Davies and Arne Kovac. Densities, spectral densities and modality. *The Annals of Statistics*, pages 1093–1136, 2004.

[35] Timothy Davis and William Hager. Dynamic supernodes in sparse Cholesky update/downdate and triangular solves. *ACM Transactions on Mathematical Software*, 35(4):1–23, 2009.

[36] Lieven De Lathauwer, Bart De Moor, and Joos Vandewalle. "A multilinear singular value decomposition". *SIAM journal on Matrix Analysis and Applications*, 21(4):1253–1278, 2000.

[37] Aurore Delaigle. Nonparametric kernel methods with errors-in-variables: Constructing estimators, computing them, and avoiding common mistakes. *Australian & New Zealand Journal of Statistics*, 56(2):105–124, 2014.

[38] Aurore Delaigle and Irène Gijbels. Estimation of integrated squared density derivatives from a contaminated sample. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 64(4):869–886, 2002.

[39] Aurore Delaigle and Peter Hall. Parametrically assisted nonparametric estimation of a density in the deconvolution problem. *Journal of the American Statistical Association*, 109(506):717–729, 2014.

[40] Aurore Delaigle and Peter Hall. Methodology for non-parametric deconvolution when the error distribution is unknown. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 78(1):231–252, 2016.

[41] Luc Devroye. The equivalence of weak, strong and complete convergence in l1 for kernel density estimates. *The Annals of Statistics*, pages 896–904, 1983.

[42] Kim-Anh Do, Peter Muller, and Feng Tang. A Bayesian mixture model for differential gene expression. *Journal of the Royal Statistical Society, Series C*, 54(3):627–44, 2005.

[43] Sophie Donnet, Vincent Rivoirard, Judith Rousseau, and Catia Scricciolo. Posterior concentration rates for empirical bayes procedures, with applications to dirichlet process mixtures. *arXiv preprint arXiv:1406.4406*, 2014.

[44] David L Donoho. Cart and best-ortho-basis: a connection. *The Annals of Statistics*, 25(5):1870–1911, 1997.

[45] David L Donoho and Iain M. Johnstone. Minimax estimation via wavelet shrinkage. *Annals of Statistics*, 26(8):879–921, 1998.

[46] David L Donoho, Iain M Johnstone, Gérard Kerkyacharian, and Dominique Picard. Wavelet shrinkage: asymptopia? *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 301–369, 1995.

[47] John Duchi, Shai Shalev-Shwartz, Yoram Singer, and Tushar Chandra. Efficient projections onto the l 1-ball for learning in high dimensions. In *Proceedings of the 25th international conference on Machine learning*, pages 272–279. ACM, 2008.

[48] Sandrine Dudoit and Mark J. Van Der Laan. *Multiple testing procedures with applications to genomics*. Springer Science & Business Media, 2007.

[49] Tarn Duong and Martin L. Hazelton. Cross-validation bandwidth matrices for multivariate kernel density estimation. *Scandinavian Journal of Statistics*, 32(3):485–506, 2005.

[50] Bradley Efron. Microarrays, empirical bayes and the two-groups model. *Statistical science*, pages 1–22, 2008.

[51] Bradley Efron. Tweedie's formula and selection bias. *Journal of the American Statistical Association*, 106(496):1602–14, 2011.

[52] Bradley Efron. Empirical bayes deconvolution estimates. *Biometrika*, 103(1):1–20, 2016.

[53] Bradley Efron and Robert Tibshirani. Using specially designed exponential families for density estimation. *The Annals of Statistics*, 24(6):2431–2461, 1996.

[54] Michael Elkin, Yuval Emek, Daniel Spielman, and Shang-Hua Teng. Lower-stretch spanning trees. *SIAM Journal on Computing*, 38(2):608–628, 2008.

[55] Michael D. Escobar and Mike West. Bayesian density estimation and inference using mixtures. *Journal of the American Statistical Association*, 90:577–88, 1995.

[56] Jianqing Fan. On the optimal rates of convergence for nonparametric deconvolution problems. *The Annals of Statistics*, pages 1257–1272, 1991.

[57] Jianqing Fan and Ja-Yong Koo. Wavelet deconvolution. *IEEE Transactions on Information Theory*, 48(3):734–747, 2002.

[58] Thomas S. Ferguson. A Bayesian analysis of some nonparametric problems. *The Annals of Statistics*, 1:209–30, 1973.

[59] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. "Applications of the lasso and grouped lasso to the estimation of sparse graphical models". Technical report, Stanford University, 2010.

[60] Matan Gavish, Boaz Nadler, and Ronald Coifman. Multiscale wavelets on trees, graphs and high dimensional data: Theory and applications to semi supervised learning. *International Conference on Machine Learning*, 27, 2010.

[61] Andrew Gelman, John B. Carlin, Hal S. Stern, and Donald B. Rubin. *Bayesian data analysis*, volume 2. Chapman & Hall/CRC Boca Raton, FL, USA, 2014.

[62] Stuart Geman and Chii-Ruey Hwang. Nonparametric maximum likelihood estimation by the method of sieves. *The Annals of Statistics*, 10(2):401–14, 1982.

[63] Subhashis Ghosal and Aad W. Van Der Vaart. Entropies and rates of convergence for maximum likelihood and bayes estimation for mixtures of normal densities. *The Annals of Statistics*, pages 1233–1263, 2001.

[64] Chris Godsil and Gordon Royle. *Algebraic Graph Theory*. Springer, 2001.

[65] I. J. Good and R. A. Gaskins. Nonparametric roughness penalties for probability densities. *Biometrika*, 58(2):255–77, 1971.

[66] Peter Hall and Alexander Meister. A ridge-parameter approach to deconvolution. *The Annals of Statistics*, 35(4):1535–1558, 2007.

[67] David Hammond, Pierre Vandergheynst, and Rémi Gribonval. Wavelets on graphs via spectral graph theory. *Applied and Computational Harmonic Analysis*, 30(2):129–150, 2011.

[68] Zaid Harchaoui and Celine Levy-Leduc. Multiple change-point estimation with a total variation penalty. *Journal of the American Statistical Association*, 105(492):1480–1493, 2010.

[69] Richard A. Harshman. "Foundations of the parafac procedure: models and conditions for an "explanatory" multimodal factor analysis". 1970.

[70] Holger Hoefling. A path algorithm for the fused lasso signal approximator. *Journal of Computational and Graphical Statistics*, 19(4):984–1006, 2010.

[71] Jan-Christian Hutter and Philippe Rigollet. Optimal rates for total variation denoising. *Annual Conference on Learning Theory*, 29:1115–1146, 2016.

[72] Hemant Ishwaran and Mahmoud Zarepour. Exact and approximate sum representations for the dirichlet process. *The Canadian Journal of Statistics/La Revue Canadienne de Statistique*, pages 269–283, 2002.

[73] Wenhua Jiang and Cun-Hui Zhang. General maximum likelihood empirical bayes estimation of normal means. *The Annals of Statistics*, 37(4):1647–1684, 2009.

[74] Nicholas A. Johnson. "A Dynamic Programming Algorithm for the Fused Lasso and L 0-Segmentation". *Journal of Computational and Graphical Statistics*, 22(2):246–260, 2013.

[75] Iain Johnstone. Gaussian estimation: sequence and wavelet models. *Unpublished manuscript*, 2011.

[76] Alexandros Karatzoglou, Xavier Amatriain, Linas Baltrunas, and Nuria Oliver. "Multiverse recommendation: n-dimensional tensor factorization for context-aware collaborative filtering". In *Proceedings of the fourth ACM conference on Recommender systems*, pages 79–86. ACM, 2010.

[77] Gérard Kerkyacharian, Dominique Picard, and Karine Tribouley. Lp adaptive density estimation. *Bernoulli*, pages 229–247, 1996.

[78] J. Kiefer and J. Wolfowitz. Consistency of the maximum likelihood estimator in the presence of infinitely many incidental parameters. *The Annals of Mathematical Statistics*, 27:887–906, 1956.

[79] S.-J. Kim, K. Koh, S. Boyd, and D. Gorinevsky. ℓ^1 trend filtering. *SIAM Reviews*, 51:339–60, 2009.

[80] Seung-Jean Kim, Kwangmoo Koh, Stephen Boyd, and Dimitry Gorinevsky. "ℓ_1 Trend Filtering". *Siam Review*, 51(2):339–360, 2009.

[81] Roger Koenker. Rebayes: empirical bayes estimation and inference in r. *R package version 0.41*, 2013.

[82] Roger Koenker and Ivan Mizera. Density estimation by total variation regularization. *Advances in Statistical Modeling and Inference, Essays in Honor of Kjell A. Doksum*, pages 613–634, 2006.

[83] Roger Koenker and Ivan Mizera. Density estimation by total variation regularization. In V Nair, editor, *Advances in Statistical Modeling and Inference: Essays in Honor of Kjell A. Doksum*, chapter 30. World Scientific, 2007.

[84] Roger Koenker and Ivan Mizera. Convex optimization, shape constraints, compound decisions, and empirical bayes rules. *Journal of the American Statistical Association*, 109(506):674–685, 2014.

[85] Eric D. Kolaczyk and Robert D. Nowak. Multiscale likelihood analysis and complexity penalized estimation. *The Annals of Statistics*, pages 500–527, 2004.

[86] Tamara G. Kolda and Brett W. Bader. "Tensor decompositions and applications". *SIAM review*, 51(3):455–500, 2009.

[87] Vladimir Kolmogorov, Thomas Pock, and Michal Rolinek. Total variation on a tree. *SIAM Journal of Imaging Sciences*, 9(2):605–636, 2016.

[88] Arne Kovac and Andrew Smith. Nonparametric regression on a graph. *Journal of Computational and Graphical Statistics*, 20(2):432–447, 2011.

[89] Pieter M. Kroonenberg. *"Applied multiway data analysis"*, volume 702. John Wiley & Sons, 2008.

[90] Loic Landrieu and Guillaume Obozinski. Cut pursuit: fast algorithms to learn piecewise constant functions on general weighted graphs. *HAL preprint hal-01306779*, 2015.

[91] Mihee Lee, Peter Hall, Haipeng Shen, James Stephen Marron, Jon Tolle, and Christina Burch. Deconvolution estimation of mixture distributions with boundaries. *Electronic journal of statistics*, 7:323, 2013.

[92] Jure Leskovec, Kevin J. Lang, Anirban Dasgupta, and Michael W. Mahoney. Community structure in large networks: Natural cluster sizes and the absence of large well-defined clusters. *Internet Mathematics*, 6(1):29–123, 2009.

[93] Kevin Lin, James Sharpnack, Alessandro Rinaldo, and Ryan J. Tibshirani. Approximate recovery in changepoint problems, from ℓ_2 estimation error rates. *arXiv preprint arXiv:1606.06746*, 2016.

[94] JK Lindsey. Comparison of probability distributions. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 38–47, 1974.

[95] JK Lindsey. Construction and comparison of statistical models. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 418–425, 1974.

[96] Li Ma. Adaptive shrinkage in pólya tree type models. *Bayesian Analysis*, 2016.

[97] Oscar-Hernan Madrid-Padilla, Nicholas G. Polson, and James G. Scott. A deconvolution path for mixtures. *arXiv preprint arXiv:1511.06750*, 2015.

[98] Oscar-Hernan Madrid-Padilla and James G. Scott. Nonparametric density estimation by histogram trend filtering. *arXiv preprint arXiv:1509.04348*, 2015.

[99] Oscar-Hernan Madrid-Padilla and James G. Scott. Tensor decomposition with generalized lasso penalties. *Journal of Computational and Graphical Statistics*, (just-accepted), 2016.

[100] Oscar-Hernan Madrid-Padilla, James G. Scott, James Sharpnack, and Ryan J. Tibshirani. The dfs fused lasso: Linear-time denoising over general graphs. *arXiv preprint arXiv:1608.03384*, 2016.

[101] Enno Mammen. Nonparametric regression under qualitative smoothness assumptions. *The Annals of Statistics*, pages 741–759, 1991.

[102] Enno Mammen and Sara van de Geer. Locally apadtive regression splines. *Annals of Statistics*, 25(1):387–413, 1997.

[103] R. Martin and Surya T. Tokdar. A nonparametric empirical Bayes framework for large-scale multiple testing. *Biostatistics*, 13(3):427–39, 2012.

[104] Ryan Martin and Surya T. Tokdar. Semiparametric inference in mixture models with predictive recursion marginal likelihood. *Biometrika*, 98(3):567–582, 2011.

[105] Omkar Muralidharan. An empirical bayes mixture method for effect size and false discovery rate estimation. *The Annals of Applied Statistics*, pages 422–438, 2010.

[106] Radford M. Neal. Mcmc using hamiltonian dynamics. *Handbook of Markov Chain Monte Carlo*, 2:113–162, 2011.

[107] Michael A. Newton. On a nonparametric recursive estimator of the mixing distribution. *Sankhyā: The Indian Journal of Statistics, Series A*, pages 306–322, 2002.

[108] Finbarr O'Sullivan. Fast computation of fully automated log-density and log-hazard estimators. *SIAM Journal on scientific and statistical computing*, 9(2):363–379, 1988.

[109] Junyang Qian and Jinzhu Jia. On pattern recovery of the fused lasso. *arXiv preprint arXiv:1211.5194*, 2012.

[110] Aaditya Ramdas and Ryan J. Tibshirani. Fast and flexible admm algorithms for trend filtering. *Journal of Computational and Graphical Statistics*, 25(3):839–858, 2016.

[111] Garvesh Raskutti, Martin Wainwright, and Bin Yu. Minimax rates of estimation for high-dimensional linear regression over ℓ_q-balls. *IEEE Transactions on Information Theory*, 57(10):6976–6994, 2011.

[112] Pradeep Ravikumar, Martin J. Wainwright, and John D. Lafferty. High-dimensional ising model selection using l_1-regularized logistic regression. *The Annals of Statistics*, 38(3):1287–1319, 2010.

[113] Alessandro Rinaldo. Properties and refinements of the fused lasso. *The Annals of Statistics*, 37(5):2922–2952, 2009.

[114] Herbert Robbins. An empirical Bayes approach to statistics. In *Proceedings of the Third Berkeley Symposium on Mathematical Statistics and Probability, 1954–1955*, volume 1, pages 157–63. University of California Press, Berkeley and Los Angeles, 1956.

[115] Herbert Robbins. The empirical bayes approach to statistical decision problems. *The Annals of Mathematical Statistics*, pages 1–20, 1964.

[116] Cristian R. Rojas and Bo Wahlberg. On change point detection using the fused lasso method. *arXiv preprint arXiv:1401.5408*, 2014.

[117] L. Rudin, S. Osher, and E. Faterni. Nonlinear total variation based noise removal algorithms. *Physica D: Nonlinear Phenomena*, 60(259–68), 1992.

[118] Leonid Rudin, Stanley Osher, and Emad Faterni. Nonlinear total variation based noise removal algorithms. *Physica D: Nonlinear Phenomena*, 60(1):259–268, 1992.

[119] Veeranjaneyulu Sadhanala, Yu-Xiang Wang, and Ryan J. Tibshirani. Total variation classes beyond 1d: Minimax rates, and the limitations of linear smoothers. *To appear, Neural Information Processing Systems*, 2016.

[120] Sylvain Sardy and Paul Tseng. Density estimation by total variation penalized likelihood driven by the sparsity 1 information criterion. *Scandinavian Journal of Statistics*, 37(2):321–337, 2010.

[121] Abhra Sarkar, Bani K Mallick, John Staudenmayer, Debdeep Pati, and Raymond J Carroll. Bayesian semiparametric density deconvolution in the presence of conditionally heteroscedastic measurement errors. *Journal of Computational and Graphical Statistics*, 23(4):1101–1125, 2014.

[122] Abhra Sarkar, Debdeep Pati, Bani K. Mallick, and Raymond J. Carroll. Bayesian semiparametric multivariate density deconvolution. *arXiv preprint arXiv:1404.6462*, 2014.

[123] Juliane Schäfer and Korbinian Strimmer. An empirical bayes approach to inferring large-scale gene association networks. *Bioinformatics*, 21(6):754–764, 2005.

[124] D.W. Scott. *Multivariate Density Estimation: Theory, Practice, and Visualization*. Wiley, 1st edition, 1992.

[125] James Sharpnack, Akshay Krishnamurthy, and Aarti Singh. Detecting activations over graphs using spanning tree wavelet bases. *International Conference on Artificial Intelligence and Statistics*, 16:536–544, 2013.

[126] David Shuman, Sunil Narang, Pascal Frossard, Antonio Ortega, and Pierre Vandergheynst. The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains. *IEEE Signal Processing Magazine*, 30(3):83–98, 2013.

[127] Bernard W. Silverman. On the estimation of a probability density function by the maximum penalized likelihood method. *The Annals of Statistics*, pages 795–810, 1982.

[128] Aarti Singh, Robert Nowak, and Robert Calderbank. Detecting weak but hierarchically-structured patterns in networks. *International Conference on Artificial Intelligence and Statistics*, 13:749–756, 2010.

[129] Dinesh Singh, Phillip G. Febbo, Kenneth Ross, Donald G. Jackson, Judith Manola, Christine Ladd, Pablo Tamayo, Andrew A. Renshaw, Anthony V. D'Amico, Jerome P. Richie, Eric S. Lander, Massimo Loda, Philip W. Kantoff, Todd R. Golub, and William R. Sellers. Gene expression correlates of clinical prostate cancer behavior. *Cancer Cell*, 1(2):203–9, 2002.

[130] Alexander Smola and Risi Kondor. Kernels and regularization on graphs. *Annual Conference on Learning Theory*, 16, 2003.

[131] John Staudenmayer, David Ruppert, and John P. Buonaccorsi. Density estimation in the presence of heteroscedastic measurement error. *Journal of the American Statistical Association*, 103(482):726–736, 2008.

[132] Leonard A. Stefanski and Raymond J. Carroll. Deconvolving kernel density estimators. *Statistics*, 21(2):169–184, 1990.

[133] Wei Sun, Junwei Lu, Han Liu, and Guang Cheng. "Provable Sparse Tensor Decomposition". *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 2016.

[134] Wesley Tansey, Alex Athey, Alex Reinhart, and James G. Scott. Multiscale spatial density smoothing: an application to large-scale radiological survey and anomaly detection. *arXiv preprint arXiv:1507.07271*, 2015.

[135] Wesley Tansey, Oluwasanmi Koyejo, Russell A. Poldrack, and James G. Scott. False discovery rate smoothing. Technical report, University of Texas at Austin, 2014. http://arxiv.org/abs/1411.6144.

[136] Wesley Tansey and James G. Scott. A fast and flexible algorithm for the graph-fused lasso. *arXiv preprint arXiv:1505.06475*, 2015.

[137] R. Tibshirani, M. Saunders, S. Rosset, J. Zhu, and K. Knight. Sparsity and smoothness via the fused lasso. *Journal of the Royal Statistical Society (Series B)*, 67:91–108, 2005.

[138] Robert Tibshirani. "Regression shrinkage and selection via the lasso". *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 267–288, 1996.

[139] Robert Tibshirani, Michael Saunders, Saharon Rosset, Ji Zhu, and Keith Knight. "Sparsity and smoothness via the fused lasso". *Journal of the Royal Statistical Society(Series B)*, 67(1):91–108, 2005.

[140] Ryan J. Tibshirani. Adaptive piecewise polynomial estimation via trend filtering. *The Annals of Statistics*, 42(1):285–323, 2014.

[141] Ryan J. Tibshirani and Jonathan Taylor. The solution path of the generalized lasso. *The Annals of Statistics*, 39:1335–71, 2011.

[142] Ryan J. Tibshirani and Jonathan Taylor. Degrees of freedom in lasso problems. *The Annals of Statistics*, 40(2):1198–1232, 2012.

[143] Surya T. Tokdar, Ryan Martin, and Jayanta K. Ghosh. Consistency of a recursive estimate of mixing distributions. *The Annals of Statistics*, pages 2502–2522, 2009.

[144] Y. H. Tsin. Some remarks on distributed depth-first search. *Information Processing Letters*, 82:173–178, 2002.

[145] Alexander Tsybakov. *Introduction to Nonparametric Estimation*. Springer, 2009.

[146] Sara Van de Geer. Estimating a regression function. *The Annals of Statistics*, pages 907–924, 1990.

[147] Stefan Wager. A geometric approach to density estimation with additive noise. *Statistica Sinica*, 2013.

[148] Abraham Wald. Note on the consistency of the maximum likelihood estimate. *The Annals of Mathematical Statistics*, pages 595–601, 1949.

[149] Yu-Xiang Wang, James Sharpnack, Alex Smola, and Ryan J. Tibshirani. Trend filtering on graphs. *Journal of Machine Learning Research*, 17(105):1–41, 2016.

[150] Rebecca M Willett and Robert D Nowak. Multiscale poisson intensity and density estimation. *Information Theory, IEEE Transactions on*, 53(9):3171–3187, 2007.

[151] Daniela M. Witten, Robert Tibshirani, and Trevor Hastie. "A penalized matrix decomposition, with applications to sparse principal components and canonical correlation analysis". *Biostatistics*, 10(3):515, 2009.

[152] Bin Yu. Assouad, Fano, and Le Cam. In *Festschrift for Lucien Le Cam*, pages 423–435. Springer, 1997.

[153] Cun-Hui Zhang. Fourier methods for estimating mixing densities and distributions. *The Annals of Statistics*, pages 806–831, 1990.

[154] Dengyong Zhou, Jiayuan Huang, and Bernhard Scholkopf. Learning from labeled and unlabeled data on a directed graph. *International Conference on Machine Learning*, 22:1036–1043, 2005.

[155] Xiaojin Zhu, Zoubin Ghahramani, and John Lafferty. Semi-supervised learning using Gaussian fields and harmonic functions. *International Conference on Machine Learning*, 20:912–919, 2003.

[156] Yunzhang Zhu. An augmented admm algorithm with application to the generalized lasso problem. *Journal of Computational and Graphical Statistics*, (just-accepted), 2015.

Vita

Oscar Hernan Madrid Padilla was born in May 18, 1991. He was raised in Honduras, in the rural areas of both regions Santa Barbara, and Olancho. Oscar is the oldest son of Alejandrina Padilla and Jose Ramon Madrid. At an early age, Oscar was encouraged by his parents to study mathematics. This led him to compete in the International Mathematical Olympiad in 2008.

Oscar earned a bachelor degree in mathematics from the Universidad de Guanajuato, in Mexico in April, 2013. He then moved to the United states, in August of 2013, to study a PhD in Statistics at The University of Texas at Austin.

To continue his statistical research, Oscar has accepted an offer to be "Neyman Visiting Assistant Professor" at the University of California, Berkeley, with expected start date of July 1, 2017.

Permanent email: oscar.madrid@utexas.edu
This dissertation was typed by the author.