Data-Adaptive Modeling using Convex Regression

Ashley Petersen

A dissertation
submitted in partial fulfillment of the
requirements for the degree of

Doctor of Philosophy

University of Washington

2016

Reading Committee:

Daniela Witten, Chair

Noah Simon, Chair

Ali Shojaie

Program Authorized to Offer Degree:
Biostatistics

University of Washington

Abstract

Data-Adaptive Modeling using Convex Regression

Ashley Petersen

Co-Chairs of the Supervisory Committee:
Dr. Daniela Witten
Departments of Biostatistics and Statistics

Dr. Noah Simon
Department of Biostatistics

The collection and storage of large amounts of data has greatly increased in recent years, which has led to a greater interest in developing methods to explore and model this data. In settings in which little is known about the relationship among variables in advance, exploratory data analysis is particularly important. Furthermore, when the number of covariates is large, traditional exploratory data analysis becomes challenging. In this setting, fitting flexible and interpretable models can be a useful tool for visualizing the conditional relationships between the covariates and outcome. In Chapter 2, we present a method that adaptively selects covariates to include in the model and for those included, models their conditional associations with the outcome as piecewise constant functions with adaptively-chosen knots. In Chapter 3, we present a related method that is useful in settings in which interactions between pairs of covariates are of particular interest. In Chapter 4, we turn our attention to a specific type of data: calcium imaging data, which measures large populations of neurons at cellular resolution in behaving animals. As a first step of analyzing calcium imaging data, two goals must be accomplished: neuron identification and calcium quantification. Through combining image segmentation, clustering, and convex regression, we are able to extract the locations of neurons in the field of view, as well as estimate the neurons'

intracellular calcium concentrations over time. This extracted data is useful in downstream analyses, which aim to provide unprecedented insight into neural activity.

TABLE OF CONTENTS

Page

LIST OF FIGURES

LIST OF TABLES

Table Number Page

ACKNOWLEDGMENTS

This dissertation would not have been possible without the support of so many. My advisors, Noah Simon and Daniela Witten, were instrumental in my success. I wish to thank Noah for his infectious enthusiasm for research, willingness to help me explore alternative solutions, and ability to distill complex concepts. I would like to thank Daniela for her absolute dedication to my success in and outside of graduate school, invaluable guidance on how to effectively communicate complicated ideas, and insightful contributions to the direction of my research. I would also like to thank my committee member, Ali Shojaie, for introducing me to research in the area of statistical learning and sharing many valuable ideas that strengthened my research. I am ever grateful to Gitana Garofalo who was immensely generous with her guidance and support throughout my studies. I would also like to thank Susanne May, Barbara McKnight, Eric Meier, Rob Schmicker, and others I have worked with at Resuscitation Outcomes Consortium. I would like to thank Brittany, Jenn, Kean Ming, and Shizhe, as well as my other classmates, for sharing in so many experiences over the past five years. I have learned so much from all of you, and had the best of times while doing so. Lastly, I would like to thank my family and friends for their unwavering support, love, and perspective.

DEDICATION

To Bryant

Chapter 1

INTRODUCTION

In recent years, it has become quick and inexpensive to collect and store large amounts of data in a number of fields. This has amplified interest in the development of statistical methods to adequately model this data. We consider the setting in which an outcome and p covariates are measured for n observations, and the goal is estimating the conditional mean of the outcome, given the covariates. When the interest is in hypothesis generation or prediction, little is often known about the data *a priori*, which makes specifying a model difficult. In particular, it may not be known which covariates are associated with the outcome, the nature of the associations, and whether there are synergistic effects between the covariates on the outcome.

The first two chapters of this dissertation focus on developing data-adaptive convex regression methodology. The proposed methods adaptively select which covariates to include in the model, determine the functional form of covariates, and specify the form of interactions between pairs of covariates. This adaptivity is crucial to being able to adequately explore and formulate hypotheses in the data-rich setting in which we live.

In Chapter 2, we consider the problem of predicting an outcome variable using p covariates measured on n independent observations, in the setting in which flexible and interpretable fits are desirable. Flexibility in a modeling approach is vital in order to capture unknown or unexpected associations in the early phases of data description and modeling. On the other hand, interpretability is always desirable in data modeling as overly complex models can limit the understanding of the scientific processes at play. Approaches in this setting typically offer either interpretability but limited flexibility (for instance, linear regression or a piecewise constant model with pre-specified knots) or flexibility but limited interpretability

(for instance, a non-parametric approach). These issues are only amplified in the high-dimensional setting for which many covariates are measured, but the number of observations is limited with $n < p$.

Previously proposed non-parametric methods often rely on a pre-specified set of basis functions. Having to choose the basis functions *a priori* (opposed to in a data-adaptive way) limits flexibility. The choice of basis functions can also limit interpretability, since for many choices of basis functions (e.g., natural cubic splines) the resulting fits can be complex and non-monotonic without clear change points. Our proposal aims to balance this trade-off between interpretability and flexibility. To do this, we propose an additive model in which the function of each predictor is estimated to be piecewise constant with a small number of adaptively-chosen knots. Our method is applicable in the high-dimensional setting, and adaptively chooses which covariates are included in the model. Our proposal is the solution to a convex optimization problem, for which a simple algorithm with guaranteed convergence to the global optimum is provided.

In Chapter 3, we consider the problem of including complex yet interpretable interactions into a regression model. For this problem, we consider the low-dimensional setting. Even when the number of covariates is small, it is challenging to specify the appropriate functional form of an interaction between a pair of covariates. Interactions are often included in models as the product of the two covariates. However, this functional form is insensitive to modeling many scientifically plausible forms of non-additivity. It is also important to be able to adaptively choose the functional form of the interaction given the limited knowledge available *a priori*.

An immensely popular method for adaptive, non-additive modeling is classification and regression trees (CART) [Breiman et al., 1984]. Though CART can easily incorporate complex interactions between covariates in an adaptive way, the interactions are chosen using a greedy approach. This leads to a model whose fit is unstable given small perturbations of the data. This high variability can compromise the scientific utility of the tree. Thus, we propose an alternative method that also incorporates interactions via partitioning the covari-

ate space into blocks. This partitioning allows a flexible functional form for the interaction, while choosing the partitions in a data-adaptive way leads to a flexible model. Unlike CART, our model is fit using a non-greedy approach by solving a convex optimization problem. This non-greedy approach translates to fits with much lower variability than those of a greedy approach, while retaining the interpretability afforded by the partitioning.

While the methods presented in Chapters 2 and 3 are applicable in a wide range of data settings, the work we present in Chapter 4 is specific to one type of data: calcium imaging data. Calcium imaging data allows us to image the activity of large populations of neurons in the brains of small animals [Ahrens et al., 2013, Dombeck et al., 2007, Huber et al., 2012, Prevedel et al., 2014]. The intracellular calcium concentrations of neurons are measured through the use of fluorescing calcium indicators [Chen et al., 2013, Looger and Griesbeck, 2012, Rochefort et al., 2008]. Calcium imaging data typically consists of tens of thousands of frames, with each frame typically consisting of hundreds of thousands of pixels. The data captures the observed fluorescence for all pixels in the image frame over time. In order to make use of this data in downstream analyses, two goals must be accomplished: (1) neuron identification and (2) calcium quantification. The aim of neuron identification is to pinpoint the locations of neurons in the field of view by determining which pixels are associated with a particular neuron. The goal of calcium quantification is to estimate the intracellular calcium concentration over time for each of the identified neurons.

Traditionally, neuron identification has been performed manually. That is, an investigator would circle the locations of neurons while viewing the calcium imaging video. This manual approach has a number of drawbacks, including subjectivity, inaccuracy, and increased infeasibility of performing it as the size of the data continues to grow. Thus we propose an automatic method to identify neurons and quantify their calcium concentrations over time. Our proposal combines image segmentation, clustering, and convex regression.

Chapter 2

FLEXIBLE AND INTERPRETABLE MODELS IN HIGH DIMENSIONS

This work is published in *Journal of Computational and Graphical Statistics* [Petersen et al., forthcoming].

In this chapter, we consider the task of predicting a response variable using p features measured on n independent observations. Approaches for this task typically offer either interpretability but limited flexibility (for instance, linear regression or a piecewise constant model with pre-specified knots) or flexibility but limited interpretability (for instance, a non-parametric approach). We propose a method that balances the trade-off between interpretability and flexibility, while also allowing for sparsity in high dimensions when $p > n$. It selects a subset of features to include in the model, and for these features it fits piecewise constant functions with knots that are chosen *adaptively* based on the data.

We now introduce some notation. We let $\boldsymbol{X}$ denote an $n \times p$ matrix, for which $\boldsymbol{x}_j$ is the jth column (feature), and for which the ith element (observation) is x_{ij}. When we consider the case of $p = 1$, we use $\boldsymbol{x}$ to denote the single feature, with ith element x_i. The response is an n-vector $\boldsymbol{y}$, with ith element y_i. To reference subvectors, we use $\boldsymbol{a}_{\mathcal{S}}$ to denote $\boldsymbol{a}$ with only the elements contained in the set $\mathcal{S}$. To reference submatrices, we use $\boldsymbol{A}_{\mathcal{S}}$ to denote $\boldsymbol{A}$ with only the columns contained in the set $\mathcal{S}$.

The rest of this chapter is organized as follows. In Section 2.1, we review related work. In Sections 2.2 and 2.3, we propose our method, present an algorithm to implement it, and examine some of its properties. Sections 2.4 and 2.5 contain the results of a simulation study and the analyses of two data sets. We consider some extensions in Section 2.6, and we close with a discussion in Section 2.7. Proofs are in Appendix A.

2.1 Previous Work

Generalized additive models (GAM) provide a flexible and general framework for modeling a response in low dimensions ($n > p$). We assume $\mathrm{E}[y_i|\boldsymbol{x}_i] = g\left(\sum_{j=1}^{p} f_j(x_{ij})\right)$, where g is a specified function and each f_j is an unknown function that we wish to estimate [Hastie and Tibshirani, 1986]. For now, we restrict our attention to the case when g is the identity function. There are a number of ways to estimate f_j — for example, we might use a smoothing or regression spline.

Flexible additive modeling in high dimensions has been an active area of research in recent years [Avalos et al., 2007, Huang et al., 2010, Li and Liu, 2014, Lin and Zhang, 2006, Sardy and Tseng, 2004, Wood et al., 2014, Zhang et al., 2011]. Recently, Ravikumar et al. [2009] proposed a high-dimensional extension of GAM called sparse additive models (SpAM), which induces sparsity in the function estimates using a standardized group lasso penalty [Simon and Tibshirani, 2012]. SpAM solves the problem

$$\underset{\boldsymbol{\beta}_j \in \mathbb{R}^d, 1 \le j \le p}{\text{minimize}} \quad \frac{1}{2n} \left\| \boldsymbol{y} - \sum_{j=1}^{p} \Psi_j \boldsymbol{\beta}_j \right\|_2^2 + \lambda \sum_{j=1}^{p} \sqrt{\frac{1}{n} \boldsymbol{\beta}_j^T \Psi_j^T \Psi_j \boldsymbol{\beta}_j}, \tag{2.1}$$

where $\boldsymbol{\beta}_j = (\beta_{j1} \cdots \beta_{jd})^T$ is a d-vector of coefficients, and $\Psi_j = [\psi_{j1} \cdots \psi_{jd}]$ is an $n \times d$ matrix of which the columns are the d basis functions used to model f_j.

Meier et al. [2009] modified (2.1) in order to obtain data-adaptive fits that can capture complex relationships if needed, but that otherwise are smooth. Their estimator is the solution to the optimization problem

$$\underset{\boldsymbol{\beta}_j \in \mathbb{R}^d, 1 \le j \le p}{\text{minimize}} \quad \frac{1}{n} \left\| \boldsymbol{y} - \sum_{j=1}^{p} \Psi_j \boldsymbol{\beta}_j \right\|_2^2 + \lambda_1 \sum_{j=1}^{p} \sqrt{\frac{1}{n} \boldsymbol{\beta}_j^T \Psi_j^T \Psi_j \boldsymbol{\beta}_j} + \lambda_2 \sum_{j=1}^{p} \sqrt{\boldsymbol{\beta}_j^T \boldsymbol{W}_j \boldsymbol{\beta}_j}, \tag{2.2}$$

where $\boldsymbol{\beta}_j = (\beta_{j1} \cdots \beta_{jd})^T$ is a d-vector of coefficients to be estimated, $\Psi_j = [\psi_{j1} \cdots \psi_{jd}]$ is an $n \times d$ matrix of which the columns are the cubic B-spline basis vectors of the jth predictor, and $\boldsymbol{W}_j$ is a $d \times d$ matrix containing the inner products of the second derivatives of the cubic B-spline basis functions. Other variations of this sparsity-smoothness penalty have also been proposed [Bühlmann and van de Geer, 2011, Meier et al., 2009].

Recently, Lou et al. [forthcoming] proposed the sparse partially linear additive model, which models a subset of the included features linearly and the remaining included features non-linearly using basis functions. The linear features do not need to be chosen *a priori*.

2.2 The Fused Lasso Additive Model

The methods described in Section 2.1 rely on a pre-specified set of basis functions. This limits flexibility, since the basis functions must be chosen *a priori* rather than in a data-adaptive way, as well as interpretability, since for many choices of basis functions (e.g., natural cubic splines) the resulting fits can be complex and non-monotonic without clear change points.

We now propose an approach to fit an additive model in which each function is estimated to be piecewise constant with a small number of knots. While this problem is easily solved when the knots are chosen *a priori*, our proposal allows the knots to be chosen adaptively.

2.2.1 The Optimization Problem

To begin, we assume that we have a single feature $\boldsymbol{x}$ that is ordered, i.e., $x_1 < x_2 < \cdots < x_n$. We wish to estimate an n-vector $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_n)$, where $\mathrm{E}[y_i|x_i] = f(x_i) = \theta_i$. The fused lasso seeks a piecewise constant estimate of $\boldsymbol{\theta}$ that involves a small number of knots, by solving the problem [Tibshirani et al., 2005]

$$\underset{\boldsymbol{\theta}\in\mathbb{R}^n}{\text{minimize}} \quad \frac{1}{2}\left\|\boldsymbol{y}-\boldsymbol{\theta}\right\|_2^2 + \lambda\left\|\boldsymbol{D}\boldsymbol{\theta}\right\|_1, \tag{2.3}$$

where $\lambda \geq 0$ is a tuning parameter and $\boldsymbol{D}$ is the discrete first derivative matrix,

$$\boldsymbol{D} = \begin{pmatrix} 1 & -1 & 0 & \cdots & 0 & 0 \\ 0 & 1 & -1 & \cdots & 0 & 0 \\ \vdots & & & & & \\ 0 & 0 & 0 & \cdots & 1 & -1 \end{pmatrix}.$$

We let $\hat{\boldsymbol{\theta}}$ denote the solution to (2.3). The ℓ_1 penalty encourages $|\hat{\theta}_{i-1} - \hat{\theta}_i|$ to equal zero when λ is large. The non-zero elements of $|\hat{\theta}_{i-1} - \hat{\theta}_i|$ correspond to knots in $\hat{\boldsymbol{\theta}}$. Consequently, $\hat{\boldsymbol{\theta}}$ provides a piecewise constant fit to the data, with adaptively-chosen knots.

Several algorithms for solving (2.3) have been proposed [Hoefling, 2010, Johnson, 2013, Liu et al., 2010].

We now consider the model $\mathrm{E}[y_i|\boldsymbol{x}_i] = \sum_{j=1}^p f_j(x_{ij}) = \sum_{j=1}^p \theta_{ji}$. We assume that each $\boldsymbol{\theta}_j$ is piecewise constant with mean zero, and we include an intercept θ_0. Let $\boldsymbol{P}_j$ denote a permutation matrix that orders the elements of the vector $\boldsymbol{x}_j$ from least to greatest. We can then solve the problem

$$\underset{\theta_0\in\mathbb{R},\boldsymbol{\theta}_j\in\mathbb{R}^n,1\leq j\leq p}{\text{minimize}} \quad \frac{1}{2}\left\|\boldsymbol{y}-\sum_{j=1}^p\boldsymbol{\theta}_j-\theta_0\mathbf{1}\right\|_2^2+\lambda\sum_{j=1}^p\|\boldsymbol{D}\boldsymbol{P}_j\boldsymbol{\theta}_j\|_1 \quad \text{subject to } \mathbf{1}^T\boldsymbol{\theta}_j=0 \quad \forall j. \tag{2.4}$$

In high dimensions, we may wish to impose sparsity on the $\boldsymbol{\theta}_j$'s, so that a given feature is completely excluded from the model. For λ sufficiently large, we do get a sparse solution in (2.4). However, this value of λ will tend to overshrink all of the estimates for $\boldsymbol{\theta}_j$, since the fused lasso penalty discourages large jumps in $\boldsymbol{\theta}_j$ (see Section 2.4 for related simulations). Therefore, we consider the modified optimization problem

$$\underset{\theta_0\in\mathbb{R},\boldsymbol{\theta}_j\in\mathbb{R}^n,1\leq j\leq p}{\text{minimize}} \quad \frac{1}{2}\left\|\boldsymbol{y}-\sum_{j=1}^p\boldsymbol{\theta}_j-\theta_0\mathbf{1}\right\|_2^2+\alpha\lambda\sum_{j=1}^p\|\boldsymbol{D}\boldsymbol{P}_j\boldsymbol{\theta}_j\|_1+(1-\alpha)\lambda\sum_{j=1}^p\|\boldsymbol{\theta}_j\|_2, \tag{2.5}$$

where $\lambda \geq 0$ and $0 \leq \alpha \leq 1$. Here, α provides a trade-off between encouraging $\hat{\boldsymbol{\theta}}_j$ to be piecewise constant, and inducing sparsity on the entire vector $\hat{\boldsymbol{\theta}}_j$ using the group lasso [Yuan and Lin, 2006]. We refer to the solution to (2.5) as the *fused lasso additive model* (FLAM).

2.2.2 An Algorithm for FLAM

Problem 2.5 is convex, and so can be solved using a general-purpose interior point method that has a per-iteration computational complexity of $\mathcal{O}(n^3p^3)$. Here we develop a much faster algorithm using block coordinate descent to solve (2.5) [Tseng, 2001]. We cycle through the features and repeatedly perform a partial minimization in a single $\boldsymbol{\theta}_j$, holding all others fixed. The solution for the partial minimization is given in Corollary 2.2.1, which follows from a more general result presented in Section 2.6.3. This corollary allows us to solve the FLAM optimization problem in $\mathcal{O}(n)$ operations per feature per iteration, by leveraging

an existing fused lasso solver that requires $\mathcal{O}(n)$ operations for an n-dimensional problem [Johnson, 2013].

Corollary 2.2.1 *The solution to the optimization problem*

$$\underset{\boldsymbol{\theta}\in\mathbb{R}^n}{\text{minimize}} \quad \frac{1}{2}\left\|\boldsymbol{y}-\boldsymbol{\theta}\right\|_2^2 + \alpha\lambda\left\|\boldsymbol{D}\boldsymbol{\theta}\right\|_1 + (1-\alpha)\lambda\left\|\boldsymbol{\theta}\right\|_2 \tag{2.6}$$

is $\left(1-\frac{(1-\alpha)\lambda}{\left\|\hat{\boldsymbol{\theta}}\right\|_2}\right)_+\hat{\boldsymbol{\theta}}$, *where* $(u)_+ = \max(u,0)$ *and* $\hat{\boldsymbol{\theta}}$ *is the solution to*

$$\underset{\boldsymbol{\theta}\in\mathbb{R}^n}{\text{minimize}} \quad \frac{1}{2}\left\|\boldsymbol{y}-\boldsymbol{\theta}\right\|_2^2 + \alpha\lambda\left\|\boldsymbol{D}\boldsymbol{\theta}\right\|_1. \tag{2.7}$$

Corollary 2.2.1 leads directly to Algorithm 1, which yields the global optimum to (2.5) [Tseng, 2001], and can be made very efficient using warm starts and active sets.

Algorithm 1 — Block Coordinate Descent for Fused Lasso Additive Model (Equation 2.5)

1. Initialize $\hat{\theta}_0 = 0$ and $\hat{\boldsymbol{\theta}}_j = \mathbf{0}$ for all $j = 1,\ldots,p$.

2. For each $j = 1,\ldots,p$, perform the following:

   a. Compute the residual $\boldsymbol{r}_j = \boldsymbol{y} - \hat{\theta}_0\mathbf{1} - \sum_{j'\neq j}\hat{\boldsymbol{\theta}}_{j'}$.

   b. Using an algorithm for the fused lasso (e.g., `flsa` on `CRAN` [Hoefling, 2013]), solve

   $$\underset{\boldsymbol{\theta}_j\in\mathbb{R}^n}{\text{minimize}} \quad \frac{1}{2}\left\|\boldsymbol{r}_j-\boldsymbol{\theta}_j\right\|_2^2 + \alpha\lambda\left\|\boldsymbol{D}\boldsymbol{P}_j\boldsymbol{\theta}_j\right\|_1.$$

   c. Compute the intercept, $\hat{\theta}_0 \leftarrow \hat{\theta}_0 + \text{mean}(\hat{\boldsymbol{\theta}}_j)$, and center, $\hat{\boldsymbol{\theta}}_j \leftarrow \hat{\boldsymbol{\theta}}_j - \text{mean}(\hat{\boldsymbol{\theta}}_j)$.

   d. Soft-scale the estimate: $\hat{\boldsymbol{\theta}}_j \leftarrow \left(1-\frac{(1-\alpha)\lambda}{\left\|\hat{\boldsymbol{\theta}}_j\right\|_2}\right)_+\hat{\boldsymbol{\theta}}_j$ where $(u)_+ = \max(u,0)$.

3. Repeat Step 2 until convergence of the objective of (2.5).

2.2.3 Connections to Other Methods

The fused lasso can be interpreted as ℓ_1 trend filtering with order $k = 0$ [Kim et al., 2009, Tibshirani, 2014]. Tibshirani [2014] showed that 0th order trend filtering is equivalent to 0th order locally adaptive regression splines, proposed by Mammen and van de Geer [1997]. Therefore, we can interpret FLAM with $\alpha = 1$ as a multi-variable extension of locally adaptive regression splines. Indeed, we illustrate FLAM's local adaptivity, or ability to produce a fit that is highly variable in one portion of the domain and constant in another, in Section 2.4.

When $\alpha = 0$, FLAM is equivalent to SpAM with $\Psi_j = \boldsymbol{I} \in \mathbb{R}^{n \times n}$ in (2.1). However, this is an impractical special case in which the design matrix does not depend on the covariates.

2.3 Properties of FLAM

We define $\tilde{\boldsymbol{y}} = \boldsymbol{y} - (\frac{1}{n}\mathbf{1}^T\boldsymbol{y})\mathbf{1}$ and $\boldsymbol{V} = [\boldsymbol{P}_1^T\boldsymbol{U} \cdots \boldsymbol{P}_p^T\boldsymbol{U}]$, where $\boldsymbol{U} \in \mathbb{R}^{n \times (n-1)}$ is the matrix obtained by centering the columns of the upper triangular matrix of 1's, and removing the nth column. The following lemma indicates that FLAM can be reparameterized in terms of the pairwise differences among the ordered elements of $\boldsymbol{\theta}_j$ (i.e., the elements of $\boldsymbol{\beta}_j = \boldsymbol{D}\boldsymbol{P}_j\boldsymbol{\theta}_j$).

Lemma 2.3.1 *Let* $\hat{\boldsymbol{\beta}} = (\hat{\boldsymbol{\beta}}_1^T \ \cdots \ \hat{\boldsymbol{\beta}}_p^T)^T$ *be the solution to*

$$\underset{\boldsymbol{\beta} \in \mathbb{R}^{(n-1)p}}{\text{minimize}} \quad \frac{1}{2}\left\|\tilde{\boldsymbol{y}} - \boldsymbol{V}\boldsymbol{\beta}\right\|_2^2 + \alpha\lambda\left\|\boldsymbol{\beta}\right\|_1 + (1-\alpha)\lambda\sum_{j=1}^{p}\left\|\boldsymbol{U}\boldsymbol{\beta}_j\right\|_2. \tag{2.8}$$

Then the solution $\hat{\theta}_0, \hat{\boldsymbol{\theta}}_1, \ldots, \hat{\boldsymbol{\theta}}_p$ *to the optimization problem (2.5) is*

$$\hat{\theta}_0 = \frac{1}{n}\mathbf{1}^T\boldsymbol{y} \text{ and } \hat{\boldsymbol{\theta}}_j = \boldsymbol{P}_j^T\boldsymbol{U}\hat{\boldsymbol{\beta}}_j \text{ for } j = 1, \ldots, p.$$

From (2.8), FLAM with $\alpha = 1$ is equivalent to solving a lasso problem. The reparametrization given in Lemma 2.3.1 will allow us to easily derive some properties of FLAM.

2.3.1 Degrees of Freedom for FLAM

Suppose that $\boldsymbol{y} \sim (\boldsymbol{\mu}, \sigma^2 \boldsymbol{I})$, and let $g(\boldsymbol{y}) = \hat{\boldsymbol{y}}$ denote the fit corresponding to some model-fitting procedure g. Then the degrees of freedom of g is defined as $\frac{1}{\sigma^2}\sum_{i=1}^{n} \mathrm{Cov}(y_i, \hat{y}_i)$ [Efron, 1986, Hastie and Tibshirani, 1990]. We now consider a modified version of FLAM, in which a small ridge penalty ensures strict convexity and enforces uniqueness of the solution,

$$\underset{\theta_0 \in \mathbb{R}, \boldsymbol{\theta}_j \in \mathbb{R}^n, 1 \le j \le p}{\text{minimize}} \quad \frac{1}{2}\left\| \boldsymbol{y} - \sum_{j=1}^{p} \boldsymbol{\theta}_j - \theta_0 \mathbf{1} \right\|_2^2 + \alpha\lambda \sum_{j=1}^{p} \|\boldsymbol{D}\boldsymbol{P}_j \boldsymbol{\theta}_j\|_1 + (1-\alpha)\lambda \sum_{j=1}^{p} \|\boldsymbol{\theta}_j\|_2 + \frac{\gamma}{2}\sum_{j=1}^{p} \|\boldsymbol{D}\boldsymbol{P}_j \boldsymbol{\theta}_j\|_2^2. \quad (2.9)$$

In (2.9), $\gamma \ge 0$ is a very small constant. For $\gamma = 0$, (2.9) is FLAM. Subject to reparameterization, (2.9) is equivalent to

$$\underset{\boldsymbol{\beta} \in \mathbb{R}^{(n-1)p}}{\text{minimize}} \quad \frac{1}{2}\|\tilde{\boldsymbol{y}} - \boldsymbol{V}\boldsymbol{\beta}\|_2^2 + \alpha\lambda\|\boldsymbol{\beta}\|_1 + (1-\alpha)\lambda \sum_{j=1}^{p} \|\boldsymbol{U}\boldsymbol{\beta}_j\|_2 + \frac{\gamma}{2}\sum_{j=1}^{p}\|\boldsymbol{\beta}_j\|_2^2. \quad (2.10)$$

We propose to estimate the degrees of freedom of FLAM as

$$\hat{df}_{FLAM} = \mathrm{Tr}\left(\boldsymbol{V}_{\mathcal{A}} \left[\boldsymbol{V}_{\mathcal{A}}^T \boldsymbol{V}_{\mathcal{A}} + (1-\alpha)\lambda S_2 + \gamma \boldsymbol{I} \right]^{-1} \boldsymbol{V}_{\mathcal{A}}^T \right) + 1, \quad (2.11)$$

where $\mathcal{A} = \{i \in \{1, \ldots, (n-1)p\} : \hat{\beta}_i \neq 0\}$ and S_2 is block diagonal with the jth block equal to $\frac{\boldsymbol{U}_{\mathcal{A}_j}^T \boldsymbol{U}_{\mathcal{A}_j}}{\|\boldsymbol{U}_{\mathcal{A}_j}\hat{\boldsymbol{\beta}}_{j\mathcal{A}_j}\|_2} - \frac{\boldsymbol{U}_{\mathcal{A}_j}^T \boldsymbol{U}_{\mathcal{A}_j}\hat{\boldsymbol{\beta}}_{j\mathcal{A}_j}\hat{\boldsymbol{\beta}}_{j\mathcal{A}_j}^T \boldsymbol{U}_{\mathcal{A}_j}^T \boldsymbol{U}_{\mathcal{A}_j}}{\|\boldsymbol{U}_{\mathcal{A}_j}\hat{\boldsymbol{\beta}}_{j\mathcal{A}_j}\|_2^3}$ with $\mathcal{A}_j = \{i \in \{1, \ldots, n-1\} : \hat{\beta}_{ji} \neq 0\}$. Note that S_2 is derived in Appendix A.2.

Proposition 2.3.2 *Assume $\boldsymbol{y} \sim MVN(\boldsymbol{\mu}, \sigma^2 \boldsymbol{I})$. Then $\hat{df}_{FLAM}$ is an unbiased estimator of the degrees of freedom of FLAM.*

When $\alpha = 1$ and $\gamma = 0$, (2.11) reduces to $\mathrm{rank}(\boldsymbol{V}_{\mathcal{A}}) + 1$, which agrees with the estimator proposed in Tibshirani and Taylor [2012]. Recall that $\hat{\boldsymbol{\beta}}_j = \boldsymbol{D}\boldsymbol{P}_j\hat{\boldsymbol{\theta}}_j$, so $\hat{\beta}_{ji}$ is the difference between $[\boldsymbol{P}_j\hat{\boldsymbol{\theta}}_j]_{i-1}$ and $[\boldsymbol{P}_j\hat{\boldsymbol{\theta}}_j]_i$. Thus non-zero elements of $\hat{\boldsymbol{\beta}}$ correspond to knots in the estimated fits. When $\boldsymbol{V}_{\mathcal{A}}$ is full rank (which only occurs when the number of knots is smaller than n), the following corollary provides a simple estimator for FLAM's degrees of freedom.

Corollary 2.3.3 *Assume $\boldsymbol{y} \sim MVN(\boldsymbol{\mu}, \sigma^2 \boldsymbol{I})$. Suppose that $\boldsymbol{V}_{\mathcal{A}}$ is full rank, and let $\alpha = 1$ in* (2.9)*. Then an unbiased estimate of the degrees of freedom of FLAM is one greater than the total number of knots across all estimated fits.*

In 1000 replicate data sets, we compare the mean of (2.11) to the mean of

$$\frac{1}{\sigma^2}\sum_{i=1}^{n}\left(\hat{y}_i-\mu_i\right)\left(y_i-\mu_i\right), \tag{2.12}$$

which provides a Monte Carlo estimate of $\frac{1}{\sigma^2}\sum_{i=1}^{n}\mathrm{Cov}(y_i,\hat{y}_i)$. Data are generated according to the high-dimensional setting of scenario 1, described in Section 2.4. Results are displayed in Figure 2.1(a).

We also propose an estimator for the degrees of freedom of SpAM (2.1). Defining $\Psi = [\Psi_1 \cdots \Psi_d]$ and $\boldsymbol{\beta} = \left(\boldsymbol{\beta}_1^T \cdots \boldsymbol{\beta}_d^T\right)^T$, we estimate SpAM's degrees of freedom as

$$\hat{df}_{SpAM} = \mathrm{Tr}\left(\Psi_{\mathcal{A}}\left[\Psi_{\mathcal{A}}^T\Psi_{\mathcal{A}}+\lambda D_2\right]^{-1}\Psi_{\mathcal{A}}^T\right), \tag{2.13}$$

where $\mathcal{A}=\{i\in\{1,\ldots,pd\}:\hat{\beta}_i\neq 0\}$ and D_2 is block diagonal with the jth block equal to $\frac{\Psi_{j\mathcal{A}_j}^T\Psi_{j\mathcal{A}_j}}{\|\Psi_{j\mathcal{A}_j}\hat{\boldsymbol{\beta}}_{j\mathcal{A}_j}\|_2}-\frac{\Psi_{j\mathcal{A}_j}^T\Psi_{j\mathcal{A}_j}\hat{\boldsymbol{\beta}}_{j\mathcal{A}_j}\hat{\boldsymbol{\beta}}_{j\mathcal{A}_j}^T\Psi_{j\mathcal{A}_j}^T\Psi_{j\mathcal{A}_j}}{\|\Psi_{j\mathcal{A}_j}\hat{\boldsymbol{\beta}}_{j\mathcal{A}_j}\|_2^3}$ with $\mathcal{A}_j=\{i\in\{1,\ldots,d\}:\hat{\beta}_{ji}\neq 0\}$.

Proposition 2.3.4 *Assume $\boldsymbol{y}\sim MVN(\boldsymbol{\mu},\sigma^2\boldsymbol{I})$. Then $\hat{df}_{SpAM}$ is an unbiased estimator of the degrees of freedom of SpAM.*

Interestingly, Ravikumar et al. [2009] proposed

$$\sum_{j=1}^{p}\sum_{k=1}^{d} I\left(\hat{\beta}_{jk}\neq 0\right) \tag{2.14}$$

as an estimator for SpAM's degrees of freedom. Figure 2.1(b) compares the means of the two estimators (2.13) and (2.14) with the mean of (2.12) across 1000 replicate data sets. In fact, we see that (2.13) is far more accurate than (2.14).

2.3.2 Range of λ that Yields Complete Sparsity

We now consider the range of λ for which $\hat{\boldsymbol{\theta}}_j=\mathbf{0}$ for $j=1,\ldots,p$, for $\alpha=1$ and $\alpha=0$.

Lemma 2.3.5 *If $\alpha=1$, then the solution to (2.5) is completely sparse if and only if $\lambda\geq\|\boldsymbol{V}^T\tilde{\boldsymbol{y}}\|_\infty$. If $\alpha=0$, then the solution is completely sparse if and only if $\lambda\geq\|\tilde{\boldsymbol{y}}\|_2$.*

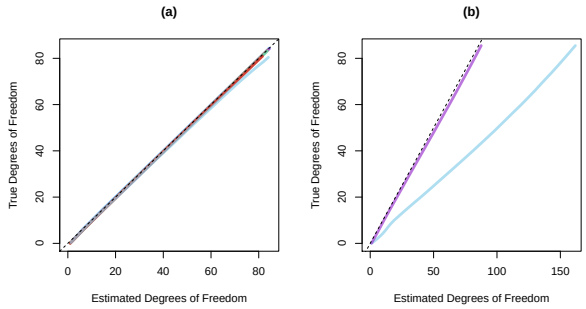


Figure 2.1: In *(a)*, we compare the degrees of freedom of FLAM calculated using (2.12) (y-axis) to the unbiased estimator (2.11) (x-axis) for $\alpha = \{0, 0.2, 0.4, 0.6, 0.8, 1\}$; each value of α is indicated by an (overlapping) colored line. In *(b)*, we compare the degrees of freedom for SpAM with $d = 3$ calculated using (2.12) (y-axis) to the estimators (2.13) (▬) and (2.14) (▬) (x-axis). In both plots, the solid lines are obtained by varying λ for FLAM or SpAM. The black dotted lines indicate $y = x$.

In Lemma 2.3.5, note that $\|\boldsymbol{V}^T\tilde{\boldsymbol{y}}\|_\infty = \max\left(g(\boldsymbol{P}_1\tilde{\boldsymbol{y}}), \ldots, g(\boldsymbol{P}_p\tilde{\boldsymbol{y}})\right)$, where $g(\boldsymbol{a}) = \max\{|a_1|, |a_1 + a_2|, \ldots, |a_1 + a_2 + \ldots + a_{n-1}|\}$. We now present a sufficient condition for the FLAM solution to be completely sparse, for any α.

Corollary 2.3.6 *For any $\alpha \in [0, 1]$, if $\lambda \geq \min\left(\frac{\|\boldsymbol{V}^T\tilde{\boldsymbol{y}}\|_\infty}{\alpha}, \frac{\|\tilde{\boldsymbol{y}}\|_2}{1-\alpha}\right)$, then the solution to (2.5) is completely sparse.*

When selecting λ for FLAM, we need never consider a value larger than that in Corollary 2.3.6.

2.3.3 Prediction Consistency

In this section, we establish prediction consistency for FLAM. For simplicity, we assume $\boldsymbol{y}$ has mean zero in this subsection. The estimated prediction error compares the predicted outcome to the best one could do if the true coefficient values $\boldsymbol{\theta}_1^0, \ldots, \boldsymbol{\theta}_p^0$ were known. Lemma 2.3.7 provides a finite sample bound for the prediction error.

Lemma 2.3.7 *Assume* $\boldsymbol{y} = \sum_{j=1}^{p} \boldsymbol{\theta}_j^0 + \boldsymbol{\epsilon}$ *with* $\boldsymbol{\epsilon} \sim MVN(\mathbf{0}, \sigma^2 \boldsymbol{I})$. *If* $\lambda \geq 2\sigma\sqrt{\frac{\log((n-1)p)}{n}}$, *then*

$$\frac{1}{n}\left\|\sum_{j=1}^{p}\left(\hat{\boldsymbol{\theta}}_j - \boldsymbol{\theta}_j^0\right)\right\|_2^2 \leq 3\lambda\sum_{j=1}^{p}\left[\alpha\left\|\boldsymbol{D}\boldsymbol{P}_j\boldsymbol{\theta}_j^0\right\|_1 + (1-\alpha)\|\boldsymbol{\theta}_j^0\|_2\right]$$

holds with probability at least $1 - \left(\frac{2}{(n-1)p} + \frac{1}{n}\right)$.

Now, assume that $\theta_{ji}^0 = f_j(x_{ij})$ where f_j has bounded variation, and all elements of $\boldsymbol{x}_j \in [a, b]$ for some a and b. Assume also that the number of non-sparse functions is bounded, i.e., $\sum_{j=1}^{p} ||f_j||_0 = K < \infty$. Together, these two assumptions imply that $\sum_{j=1}^{p} \|\boldsymbol{D}\boldsymbol{P}_j\boldsymbol{\theta}_j^0\|_1 = O(1)$, and that $\sum_{j=1}^{p} \|\boldsymbol{\theta}_j^0\|_2 = O(\sqrt{n})$. Thus, FLAM is prediction consistent provided that $(1-\alpha) = o((\log((n-1)p))^{-1/2})$, and $\lambda = 2\sigma\sqrt{\frac{\log((n-1)p)}{n}}$.

Our theory suggests that the necessary α to achieve prediction consistency converges to 1 as n and p grow. However, we have found in practice that $\alpha < 1$ is desirable in finite samples with a large number of covariates (see Section 2.4 for simulations).

2.4 Simulations

We compare the performance of FLAM to two competitors: GAM using smoothing splines (implemented with the `R` package `gam` [Hastie, 2013]), and SpAM with basis vectors corresponding to a natural cubic spline with $d-1$ non-boundary knots at equally spaced quantiles of $\boldsymbol{x}_j$ (implemented with the `R` package `SAM` [Zhao et al., 2014]). Data are generated according to $y_i = \sum_{j=1}^{p} f_j(x_{ij}) + \epsilon_i$ with $\epsilon_i \overset{iid}{\sim} N(0,1)$, $x_{ij} \overset{iid}{\sim}$ Uniform$[-2.5, 2.5]$, and $p = 4$. We consider four scenarios, displayed in Figure 2.2:

Scenario 1: All f_j are piecewise constant functions (Figure 2.2(a)).

Scenario 2: All f_j are smooth functions (Figure 2.2(b)). These are the exact functions used for the simulations in the original SpAM paper [Ravikumar et al., 2009].

Scenario 3: Two of the f_j are piecewise constant functions and the other two f_j are smooth functions (Figure 2.2(c)). This is a compromise between scenarios 1 and 2.

Scenario 4: All f_j are functions that are constant in some areas of the domain and highly variable in other areas of the domain (Figure 2.2(d)).

All functions are constructed such that $\int_{-2.5}^{2.5} f_j = 0$ and $\int_{-2.5}^{2.5} f_j^2 = 1$. We refer to scenarios 1-4 as the *low-dimensional setting*. Additionally, we refer to the same scenarios with the addition of 96 noise functions (i.e., $f_5, \ldots, f_{100} = 0$) as the *high-dimensional setting*. We note that GAM can only be applied in the low-dimensional setting ($n > p$).

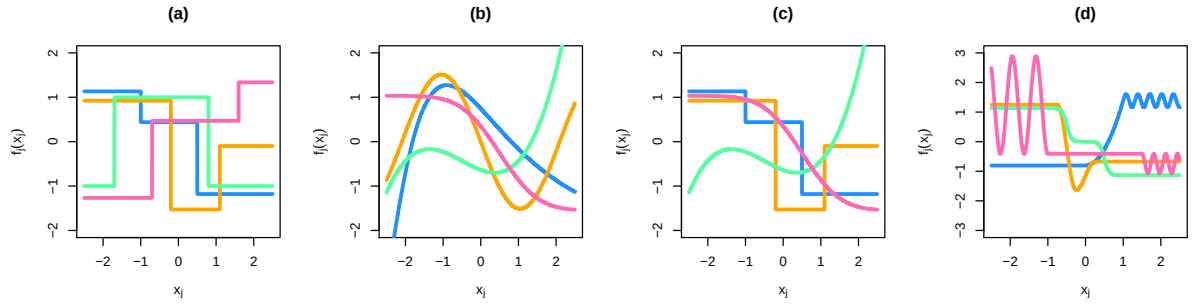


Figure 2.2: Functions used to generate data in: *(a)* scenario 1, *(b)* scenario 2, *(c)* scenario 3, and *(d)* scenario 4.

For each scenario, we generate training, test, and validation sets, each with $n = 100$. Functions are fit on the training set, and mean squared error $\left(\text{MSE}; \frac{1}{n}\|\boldsymbol{y} - \hat{\boldsymbol{y}}\|_2^2\right)$ is evaluated on the test set. For FLAM, we fix $\alpha = 0.5, 0.75,$ or 1 and consider a range of λ. For GAM, we consider a range of degrees of freedom for each smoothing spline, from 1 (just a linear fit) to n/p. For SpAM, we fix $d = 3, 6,$ or 10 and consider a range of λ.

We evaluate each method's performance as a function of its degrees of freedom. For GAM, the total degrees of freedom is p multiplied by the degrees of freedom for each covariate's smoothing spline, plus one degree of freedom for the intercept. For FLAM and SpAM, the degrees of freedom are estimated using (2.11) and (2.13), respectively.

Figure 2.3 displays the test set MSE versus total degrees of freedom for the three methods. FLAM achieves the lowest test set MSE across all scenarios except in scenario 2 where all f_j

are smooth. GAM performs comparably to SpAM in scenario 3 without noise functions. As expected, FLAM with $\alpha = 1$ outperforms FLAM with $\alpha < 1$ in the scenarios without noise functions, as no additional sparsity is needed. In general, $\alpha < 1$ is preferred in the scenarios with noise functions, with the exception of scenario 4. We discuss this discrepancy below.

Additionally, we summarize performance for the *optimal* tuning parameter, defined as the tuning parameter corresponding to the minimum test set MSE. We calculate the validation set MSE for the training set fit corresponding to the optimal tuning parameter, as well as the parameter fit ($\sum_{j=1}^{p} \|\boldsymbol{\theta}_j - \hat{\boldsymbol{\theta}}_j\|_2^2$), sparsity, and degrees of freedom (Table 2.1). Once again, FLAM performs best in all scenarios except when all f_j are smooth (scenario 2), with the best performance corresponding to FLAM with $\alpha = 1$ in the low-dimensional setting and $\alpha = 0.75$ in the high-dimensional setting. In scenario 4 with noise functions, FLAM with $\alpha = 1$ is able to achieve comparable sparsity to FLAM with $\alpha < 1$. This explains the optimal performance of FLAM with $\alpha = 1$ in this setting (Figure 2.3).

A strength of FLAM is its local adaptivity, or ability to produce a fit that is highly variable in one portion of the domain and constant in another. We can see this qualitatively by examining the function fits for scenario 4 with 96 noise functions. In Figure 2.4, we plot the fits corresponding to the the optimal tuning parameter (as defined above) for the truly non-zero functions, across 25 replicate data sets. In general, FLAM more adeptly fits both the constant and highly variable regions of the functions, relative to SpAM. SpAM's local adaptivity is limited due to the types of penalties imposed in (2.1) — λ encourages the entire $\hat{f}_j$ to be zero, while d controls the amount of flexibility in each $\hat{f}_j$. Having fewer basis functions (i.e., small d) results in less variable function fits, while a large d can produce highly variable function fits. However, the amount of variability cannot be varied greatly over the domain of the function fit, unless basis functions are specifically chosen for this purpose *a priori*.

2.5 Data Application

2.5.1 Predictors of a Country's Happiness

We now consider whether wealth is associated with happiness, by estimating the conditional relationships between a country-level happiness index and gross national income, as well as 11 other country-level predictors. The happiness index is the average of Cantril Scale [Cantril, 1965] responses of approximately 3000 residents in each country obtained in Gallup World Polls from 2010-2012, publicly available from the United Nations (UN) 2013 World Happiness Report [Helliwell et al., 2013]. The predictors are publicly available through the UN Human Development Reports and the World Bank Development Indicators [UNDP, 2012, World Bank Group, 2012]. They are from 2012 data or the closest year prior.

We consider 10 splits of the 109 countries with complete data into training and test sets. We compare FLAM to GAM with an identity link and smoothing splines using the `R` package `gam` [Hastie, 2013]. Tuning parameters are chosen using 10-fold CV in the training set. The estimated fits are shown in Figure 2.5. Both methods provide a large improvement in average test set MSE across 10 splits of the data compared to the intercept-only model (FLAM: 0.367; GAM: 0.308; intercept-only: 1.19). FLAM's estimated fits are both intuitive and fairly similar across the different splits of data. Conditional on the other predictors, FLAM estimates that increased gross national income is associated with increased happiness, up to a certain level of income. Beyond that, happiness is constant. We were quite surprised to see that GAM finds a *negative* conditional association between a country's happiness index and the number of scientific journal publications. Reassuringly, FLAM found no such association.

2.5.2 Classification based on Gene Expression

In this section, we apply FLAM with logistic loss (to be discussed in Section 2.6), in order to perform classification using gene expression measurements. The data sets we consider are:

1. *Autism* [Alter et al., 2011]: 1498 gene expression measurements from peripheral blood

lymphocytes sampled from 82 children with autism and 60 controls, publicly available from GEO at accession number GDS4431 [Barrett et al., 2007].

2. *Lung S* [Spira et al., 2007]: 22,283 gene expression measurements from large airway epithelial cells sampled from 97 smokers with lung cancer and 90 smokers without lung cancer, available from GEO at accession number GDS2771.

3. *Lung NS* [Lu et al., 2010]: 54,675 gene expression measurements from 60 pairs of tumor and adjacent normal lung tissue from non-smoking women with non-small cell lung carcinoma, available from GEO at accession number GDS3837.

We consider only the 2000 genes with the largest variance in the *Lung S* and *Lung NS* data sets. We compare the performances of FLAM to SpAM and ℓ_1-penalized logistic regression over 30 splits of the data into training and test sets, after standardizing each gene to have mean zero and variance one in the training set. We choose the tuning parameters using 10-fold CV in the training set and calculate the misclassification rate in the test set.

Test error and sparsity (the percent of genes not used in the classifier) are shown in Figure 2.6. FLAM has the same or better predictive performance on average as SpAM, but uses a less sparse classifier. However, lasso's performance is comparable to FLAM and SpAM, which indicates that the sample size may be too small to successfully model non-linear relationships in these three data sets.

For one split of the Lung S data, Figure 2.6 displays the estimated fits from FLAM, SpAM, and lasso for six genes. These six genes were selected because they were among the 15 with the highest-variance fits for both FLAM and SpAM. Note that the since the genes estimated to have a non-zero relationship with the response differed for each method, the conditional fits shown for a particular gene are not directly comparable across methods.

2.6 Extensions to FLAM

We now consider the general optimization problem

$$\underset{\boldsymbol{\theta}\in\mathbb{R}^{np}}{\text{minimize}} \quad \ell(\boldsymbol{\theta}) + \lambda \sum_{j=1}^{p} Q_j(\boldsymbol{\theta}_j), \tag{2.15}$$

where $\boldsymbol{\theta} = (\boldsymbol{\theta}_1^T \ \cdots \ \boldsymbol{\theta}_p^T)^T$, $\ell : \mathbb{R}^{np} \to \mathbb{R}$ is a differentiable, convex loss function with Lipschitz continuous gradient $\dot{\ell}$, and $Q_j(\cdot)$ is a convex penalty function. We denote the Hessian of ℓ as $\ddot{\ell}$. Thus far we have considered (2.15) for squared error loss, i.e., $\ell(\boldsymbol{\theta}) = \frac{1}{2}\left\|\boldsymbol{y} - \sum_{j=1}^{p}\boldsymbol{\theta}_j - \theta_0\mathbf{1}\right\|_2^2$, and $Q_j(\boldsymbol{\theta}_j) = \alpha\left\|\boldsymbol{D}\boldsymbol{P}_j\boldsymbol{\theta}_j\right\|_1 + (1-\alpha)\left\|\boldsymbol{\theta}_j\right\|_2$ for $\alpha \in [0,1]$. In this section, we discuss extensions of FLAM to (1) other losses $\ell(\boldsymbol{\theta})$ and (2) other penalties Q_j.

2.6.1 A General Algorithm

Generalized gradient descent (GGD) can be used to solve (2.15) [Beck and Teboulle, 2009]. That is, (2.15) can be solved by choosing an initial $\hat{\boldsymbol{\theta}}^0$ and continually updating

$$\hat{\boldsymbol{\theta}}^k = \underset{\boldsymbol{\theta}\in\mathbb{R}^{np}}{\text{argmin}} \quad \frac{L}{2}\left\|\boldsymbol{\theta} - \hat{\boldsymbol{\theta}}^{k-1} + \frac{1}{L}\dot{\ell}(\hat{\boldsymbol{\theta}}^{k-1})\right\|_2^2 + \lambda\sum_{j=1}^{p} Q_j(\boldsymbol{\theta}_j) \tag{2.16}$$

until convergence of the objective of (2.15), where $L \in \mathbb{R}$ is such that $\ddot{\ell}(\cdot) \preceq L\boldsymbol{I}$. Equation 2.16 is separable in $\boldsymbol{\theta}_j$, so the features can be updated in parallel during each iteration of GGD (in contrast to coordinate descent, in which the features are updated sequentially). In the special case of (2.15) given in (2.5), GGD provides an alternative to Algorithm 1. Details are omitted in the interest of brevity.

2.6.2 Generalized FLAM

We now consider the model $\mathrm{E}[y_i|\boldsymbol{x}_i] = g\left(\sum_{j=1}^{p} f_j(x_{ij})\right)$, where $g(\cdot)$ is a specified function. For instance, in the case of a binary response, we can consider the mean model $\mathrm{E}[y_i|\boldsymbol{x}_i] = \mathrm{expit}(\theta_0 + \sum_{j=1}^{p}\theta_{ji})$, define $\boldsymbol{\theta} = (\theta_0\mathbf{1}^T \ \boldsymbol{\theta}_1^T \ \cdots \ \boldsymbol{\theta}_p^T)^T$, and take the loss to be logistic,

$$\ell(\boldsymbol{\theta}) = -\boldsymbol{y}^T\left((\mathbf{1}_{p+1}^T \otimes \boldsymbol{I}_n)\boldsymbol{\theta}\right) + \mathbf{1}^T\log\left(1 + \exp\left((\mathbf{1}_{p+1}^T \otimes \boldsymbol{I}_n)\boldsymbol{\theta}\right)\right).$$

We then solve (2.15) by continually updating (2.16), which amounts to the updates

$$\hat{\theta}_0^k = \hat{\theta}_0^{k-1} - \frac{4}{n(p+1)}\left[\text{expit}\left(\hat{\theta}_0^{k-1}\mathbf{1} + \sum_{j=1}^{p}\hat{\boldsymbol{\theta}}_j^{k-1}\right) - \boldsymbol{y}\right]^T \mathbf{1}$$

$$\hat{\boldsymbol{\theta}}_j^k = \underset{\boldsymbol{\theta}_j \in \mathbb{R}^n}{\text{argmin}} \quad \frac{p+1}{8}\left\|\boldsymbol{\theta}_j - \hat{\boldsymbol{\theta}}_j^{k-1} + \frac{4}{p+1}\left[\text{expit}\left(\hat{\theta}_0^{k-1}\mathbf{1} + \sum_{j=1}^{p}\hat{\boldsymbol{\theta}}_j^{k-1}\right) - \boldsymbol{y}\right]\right\|_2^2 + \lambda Q_j(\boldsymbol{\theta}_j) \quad (2.17)$$

for $j = 1, \ldots, p$. The solution of (2.17) follows from Corollary 2.2.1 when $Q_j(\boldsymbol{\theta}_j) = \alpha \left\|\boldsymbol{D}\boldsymbol{P}_j\boldsymbol{\theta}_j\right\|_1 + (1-\alpha)\left\|\boldsymbol{\theta}_j\right\|_2$ for $\alpha \in [0,1]$. We now consider (2.17) with a more general form of Q_j.

2.6.3 FLAM with an Alternative Penalty

Thus far, we have seen that (2.15) can be solved by repeatedly solving a problem of the form (2.16). When $Q_j(\boldsymbol{\theta}_j) = \alpha \left\|\boldsymbol{D}\boldsymbol{P}_j\boldsymbol{\theta}_j\right\|_1 + (1-\alpha)\left\|\boldsymbol{\theta}_j\right\|_2$ for $\alpha \in [0,1]$, the solution to (2.16) follows from Corollary 2.2.1. Lemma 2.6.1 generalizes this Corollary to other forms of the penalty Q_j.

Lemma 2.6.1 *For any norm $\|\cdot\|$, and any matrix $\boldsymbol{B}$ with n columns, the solution to*

$$\underset{\boldsymbol{\theta}\in\mathbb{R}^n}{\text{minimize}} \quad \frac{1}{2}\left\|\boldsymbol{y} - \boldsymbol{\theta}\right\|_2^2 + \alpha\lambda\left\|\boldsymbol{B}\boldsymbol{\theta}\right\| + (1-\alpha)\lambda\left\|\boldsymbol{\theta}\right\|_2 \quad (2.18)$$

is $\left(1 - \frac{(1-\alpha)\lambda}{\left\|\hat{\boldsymbol{\theta}}\right\|_2}\right)_+ \hat{\boldsymbol{\theta}}$, *where* $(u)_+ = \max(u, 0)$ *and* $\hat{\boldsymbol{\theta}}$ *is the solution to*

$$\underset{\boldsymbol{\theta}\in\mathbb{R}^n}{\text{minimize}} \quad \frac{1}{2}\left\|\boldsymbol{y} - \boldsymbol{\theta}\right\|_2^2 + \alpha\lambda\left\|\boldsymbol{B}\boldsymbol{\theta}\right\|. \quad (2.19)$$

2.6.4 Simulations for Generalized FLAM using Logistic Loss

We now present simulation results of FLAM for logistic loss and $Q_j(\boldsymbol{\theta}_j) = \lambda\left\|\boldsymbol{D}\boldsymbol{P}_j\boldsymbol{\theta}_j\right\|_1$. Data are generated according to $y_i \overset{iid}{\sim} \text{Bernoulli}(\text{expit}[f_1(x_{i1}) + f_2(x_{i2})])$ with $x_{ij} \overset{iid}{\sim} \text{Uniform}[-2.5, 2.5]$, where f_1 and f_2 are taken to be two of the piecewise constant functions considered previously (Figure 2.2(a)). Figure 2.7(a) shows the expectation of y_i as a function of x_1 and x_2. For each replication, we generate training and test sets with $n = 100$. We choose the value λ corresponding to the minimum test set MSE. The estimated expectation of y_i averaged over 25 data replicates is displayed in Figure 2.7(b). It closely mirrors Figure 2.7(a).

2.7 Discussion

We have presented the fused lasso additive model, a flexible yet interpretable framework for prediction, for which the estimated fits are piecewise constant with data-adaptive knots.

While the ℓ_1 penalty in (2.5) limits the number of knots in the fits, it also shrinks the magnitude of jumps where knots do occur. However, the resulting shrinkage can easily be addressed by debiasing the fit. That is, FLAM can be used to identify the knots; then the piecewise constant model can be refit using standard linear regression with the appropriate basis functions for the known knots.

While the piecewise constant framework has much flexibility, a large number of knots are needed to accommodate trends with large slopes. Piecewise linear fits are more suited to this type of relationship. The problem of estimating piecewise trends of any order between a single predictor and response has been previously explored [Kim et al., 2009, Tibshirani, 2014]. We leave the extension of FLAM to this setting to future work.

The `R` package `flam` is available on `CRAN`. The `R` package `Shiny` [RStudio and Inc., 2014] was used to develop interactive web applications demonstrating the performance of FLAM on simulated and user-uploaded data (`ajpete.shinyapps.io/FLAM` and `ajpete.shinyapps.io/FLAMdata`).

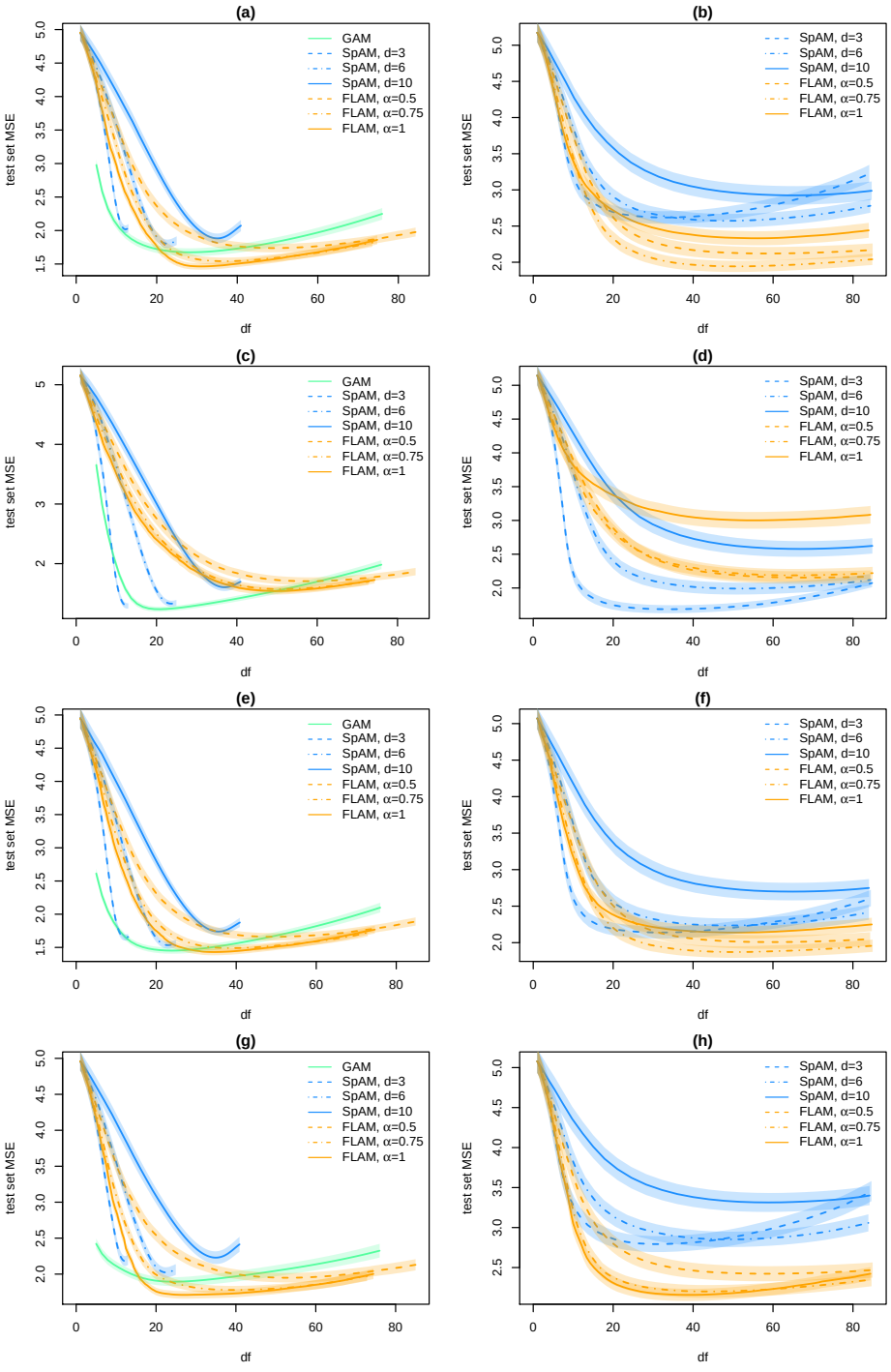


Figure 2.3: Mean test set MSE plotted by degrees of freedom (df) for: *(a)* scenario 1, *(b)* scenario 1 with noise functions, *(c)* scenario 2, *(d)* scenario 2 with noise functions, *(e)* scenario 3, *(f)* scenario 3 with noise functions, *(g)* scenario 4, and *(h)* scenario 4 with noise functions. Shaded bands indicate point-wise 95% confidence intervals over the 100 replicate data sets. GAM is only applicable in the low-dimensional setting.

Table 2.1: Results on the validation data, with the tuning parameter value chosen based on test set MSE. Mean (standard error) across 100 replicate data sets are shown. Parameter fit is defined as $\sum_{j=1}^{p} \left\| \boldsymbol{\theta}_j - \hat{\boldsymbol{\theta}}_j \right\|_2^2$. Note that GAM can only be applied in the low-dimensional setting. All $\hat{f}_j$ were non-zero for all methods in the low-dimensional setting.

	Low-dimensional			*High-dimensional*			
	MSE	Parameter fit	Degrees of freedom	MSE	Parameter fit	Proportion $\hat{f}_j$ non-zero	Degrees of freedom
Scenario 1							
FLAM, $\alpha = 0.5$	1.73 (0.03)	75.6 (1.8)	51.0 (1.1)	2.11 (0.05)	113.0 (3.3)	0.20 (0.01)	61.6 (1.5)
FLAM, $\alpha = 0.75$	1.52 (0.03)	55.4 (1.5)	39.0 (1.0)	1.92 (0.04)	95.0 (2.9)	0.23 (0.01)	54.0 (1.5)
FLAM, $\alpha = 1$	1.45 (0.02)	48.2 (1.4)	32.7 (0.8)	2.30 (0.06)	132.7 (3.4)	0.35 (0.01)	58.8 (1.9)
GAM	1.67 (0.02)	65.1 (1.3)	28.7 (0.9)				
SpAM, $d = 3$	2.02 (0.03)	107.5 (1.4)	12.2 (0.1)	2.54 (0.05)	162.0 (2.6)	0.23 (0.01)	36.7 (1.3)
SpAM, $d = 6$	1.79 (0.03)	110.0 (3.0)	22.7 (0.2)	2.52 (0.05)	168.2 (3.3)	0.23 (0.01)	50.7 (1.5)
SpAM, $d = 10$	1.85 (0.03)	130.5 (3.3)	35.4 (0.3)	2.91 (0.06)	208.9 (4.1)	0.24 (0.01)	66.5 (1.6)
Scenario 2							
FLAM, $\alpha = 0.5$	1.66 (0.03)	69.1 (1.9)	60.0 (1.1)	2.14 (0.05)	112.6 (3.4)	0.22 (0.01)	71.4 (1.5)
FLAM, $\alpha = 0.75$	1.51 (0.02)	56.1 (1.4)	52.9 (0.9)	2.17 (0.05)	115.8 (3.4)	0.27 (0.01)	66.5 (1.6)
FLAM, $\alpha = 1$	1.46 (0.02)	52.8 (1.3)	50.2 (0.9)	2.94 (0.06)	192.5 (4.1)	0.36 (0.01)	60.9 (2.3)
GAM	1.19 (0.02)	23.3 (0.7)	21.6 (0.4)				
SpAM, $d = 3$	1.21 (0.02)	32.9 (1.3)	12.5 (0.1)	1.65 (0.03)	70.3 (2.3)	0.23 (0.01)	37.4 (1.5)
SpAM, $d = 6$	1.27 (0.02)	54.6 (2.1)	23.6 (0.1)	1.95 (0.05)	109.7 (3.6)	0.23 (0.01)	53.0 (1.5)
SpAM, $d = 10$	1.50 (0.03)	84.1 (2.4)	37.1 (0.2)	2.55 (0.06)	168.3 (4.6)	0.25 (0.01)	68.6 (1.5)
Scenario 3							
FLAM, $\alpha = 0.5$	1.63 (0.03)	66.8 (1.6)	51.8 (1.1)	1.98 (0.04)	101.2 (3.3)	0.20 (0.01)	61.6 (1.5)
FLAM, $\alpha = 0.75$	1.45 (0.02)	49.2 (1.3)	40.2 (0.9)	1.84 (0.04)	88.1 (2.7)	0.24 (0.01)	54.9 (1.6)
FLAM, $\alpha = 1$	1.38 (0.02)	43.7 (1.3)	36.5 (0.8)	2.12 (0.04)	115.2 (3.1)	0.31 (0.01)	55.0 (2.1)
GAM	1.44 (0.02)	44.2 (1.0)	24.1 (0.7)				
SpAM, $d = 3$	1.62 (0.02)	69.0 (1.4)	12.2 (0.1)	2.09 (0.04)	115.3 (2.7)	0.23 (0.01)	35.1 (1.3)
SpAM, $d = 6$	1.51 (0.02)	73.3 (2.4)	22.9 (0.1)	2.18 (0.04)	134.4 (3.5)	0.23 (0.01)	50.6 (1.3)
SpAM, $d = 10$	1.68 (0.03)	103.7 (3.1)	35.7 (0.2)	2.68 (0.05)	188.3 (4.2)	0.24 (0.01)	66.7 (1.6)
Scenario 4							
FLAM, $\alpha = 0.5$	1.91 (0.04)	91.4 (1.9)	55.6 (1.3)	2.38 (0.05)	137.1 (3.2)	0.21 (0.01)	61.5 (1.9)
FLAM, $\alpha = 0.75$	1.73 (0.03)	74.5 (1.6)	42.9 (1.2)	2.15 (0.04)	115.4 (2.6)	0.21 (0.01)	45.8 (1.7)
FLAM, $\alpha = 1$	1.64 (0.03)	67.2 (1.5)	33.8 (1.3)	2.13 (0.03)	112.1 (2.4)	0.25 (0.01)	43.1 (1.6)
GAM	1.88 (0.03)	82.2 (1.4)	28.5 (1.6)				
SpAM, $d = 3$	2.15 (0.03)	121.2 (2.0)	12.2 (0.1)	2.75 (0.05)	176.2 (3.2)	0.21 (0.01)	32.9 (1.2)
SpAM, $d = 6$	2.01 (0.03)	120.6 (2.6)	22.6 (0.2)	2.78 (0.05)	187.8 (3.9)	0.24 (0.01)	51.3 (1.7)
SpAM, $d = 10$	2.19 (0.04)	157.1 (3.5)	35.0 (0.3)	3.23 (0.06)	237.4 (4.4)	0.22 (0.01)	60.0 (1.8)

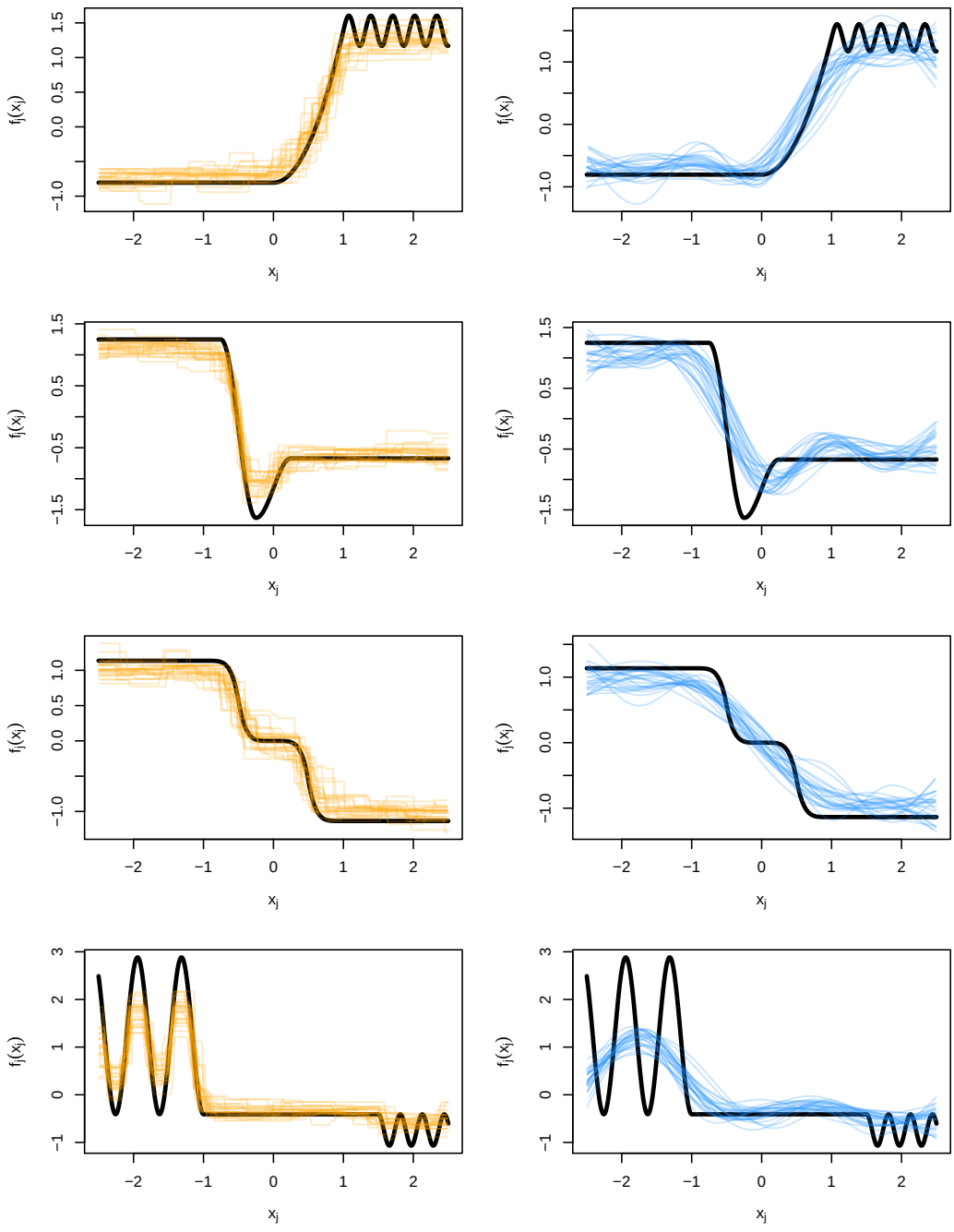


Figure 2.4: We compare the fits of FLAM (▬) and SpAM (▬) with the true underlying functions (▬) generated according to scenario 4. In each panel, 25 curves are shown. Each curve corresponds to a validation set fit with the optimal tuning parameter value, over one simulated data set.

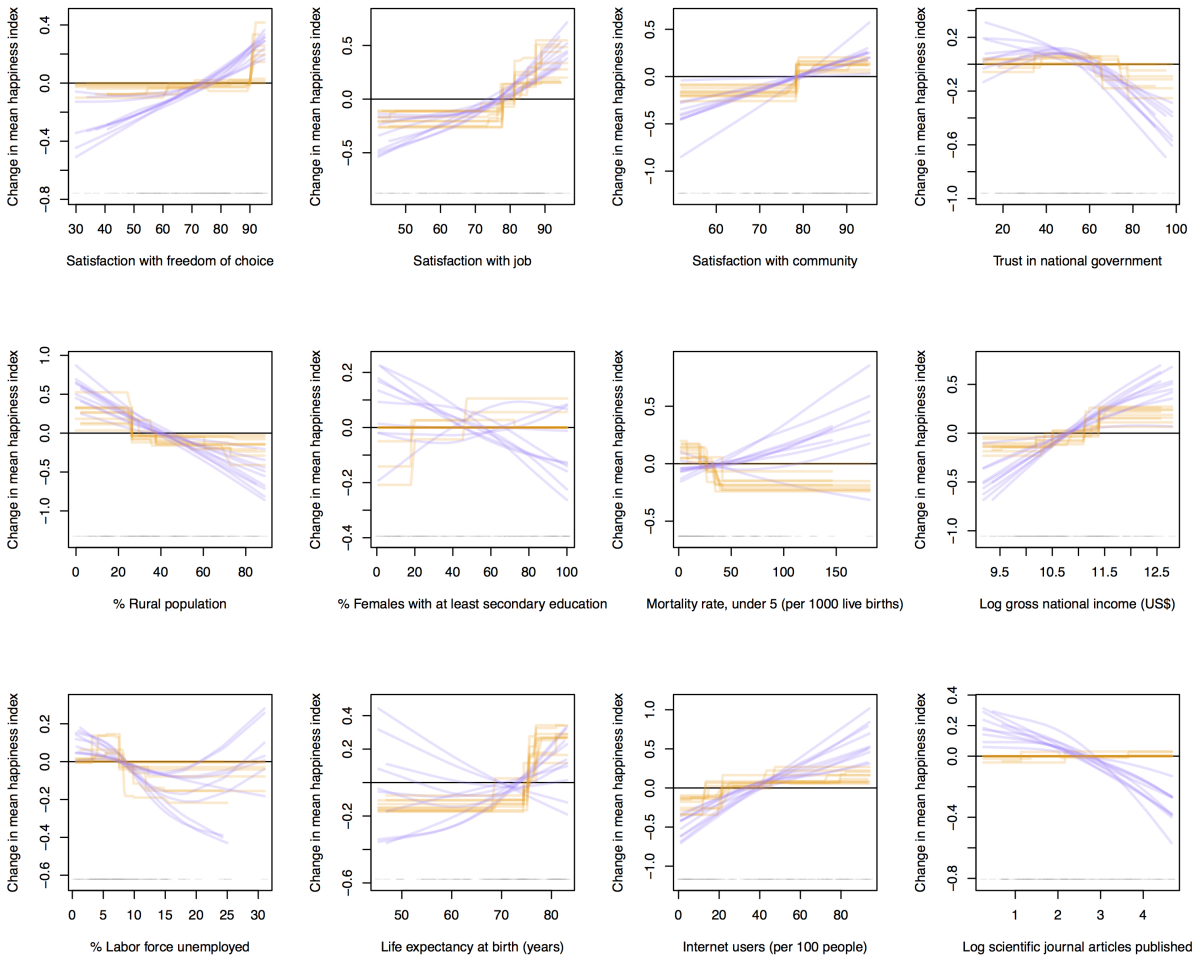


Figure 2.5: Conditional associations between the happiness index for a country and twelve country-level predictors were estimated using FLAM (▬) and GAM (▬). Ten fits for each method were obtained by repeatedly splitting the data into training and test sets. The gray bar at the bottom of each plot indicates the distribution of that predictor.

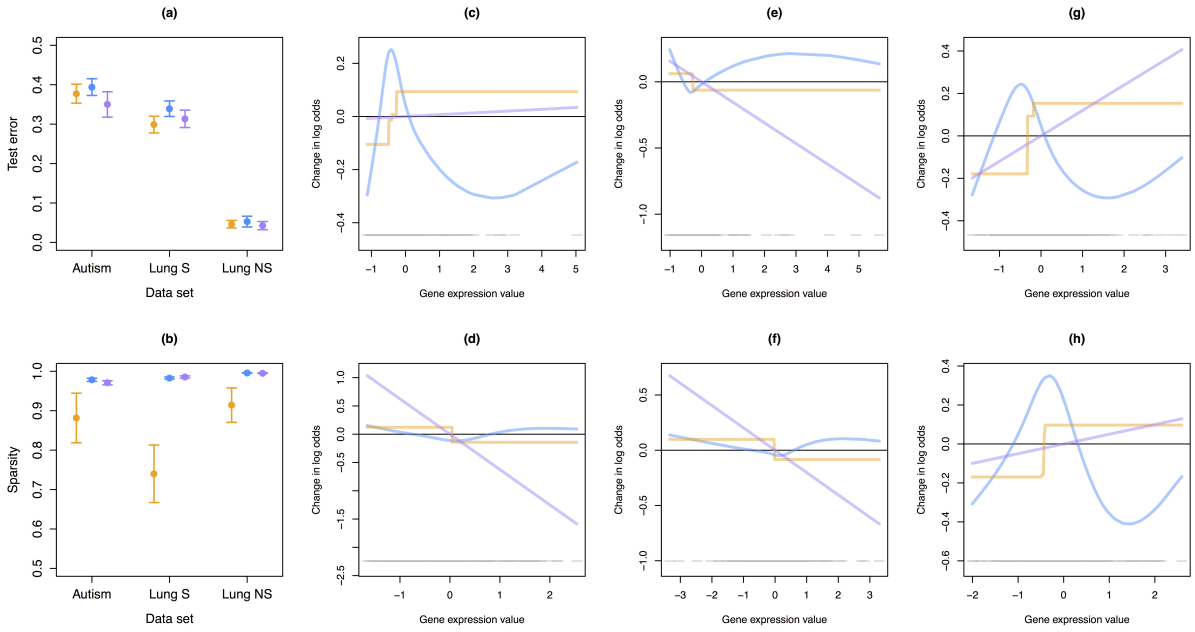


Figure 2.6: We compare the classification performance of FLAM (▬) to SpAM (▬) and lasso (▬) in terms of *(a)* test set error and *(b)* sparsity for three gene expression data sets. Both plots show mean estimates with 95% confidence intervals, which are calculated using 30 splits of the data into training and test sets. In *(c)*-*(h)*, for six genes we show the fits from FLAM (▬), SpAM (▬), and lasso (▬), which were estimated from one split of the Lung S data. The gray bar at the bottom of each plot indicates the distribution of that predictor.

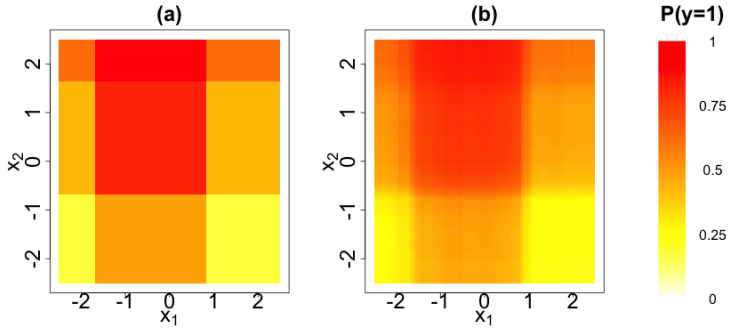


Figure 2.7: We compare *(a)* the true expectation of y_i to *(b)* the estimated expectation of y_i (averaged over 25 data replicates) obtained from FLAM with logistic loss for combinations of the two predictors, x_1 and x_2.

Chapter 3

ADAPTIVE MODELS TO CAPTURE INTERACTIONS USING PARTITIONING

This work is published in *Journal of Machine Learning Research* [Petersen et al., 2016].

Classification and regression trees (CART) are immensely popular for flexible and non-additive predictive modeling, despite the fact that they date back more than thirty years [Breiman et al., 1984]. The trees are fit using a two-stage process in which the tree is first greedily "grown" to some maximum size, and then "pruned" to avoid overfitting. The final tree with K terminal nodes can be visually displayed as a decision tree with $K-1$ splits, or equivalently as K disjoint boxes that completely partition the covariate space. CART has stood the test of time, because its output is highly interpretable and it can easily incorporate complex non-additive relationships between features. However, it is a greedy procedure, and a small perturbation of the data can produce a very different tree. The high variability of the fitted values can compromise the scientific utility of the tree, as well as the tree's prediction accuracy on test data. While an ensemble approach, like random forests, can reduce CART's variability, this comes at the expense of interpretability [Breiman, 2001].

Two other well-known methods for flexible and non-additive predictive modeling are multivariate adaptive regression splines (MARS) [Friedman, 1991] and thin-plate splines (TPS) [Duchon, 1977]. The MARS fit is a weighted sum of basis functions, which are greedily chosen and some of which involve pairs of features. TPS fits the observed data, regularized by smoothness penalties. In the case of two covariates x_1 and x_2 and a response y, the TPS fit is the solution to

$$\underset{f}{\text{minimize}} \quad \sum_{i=1}^{n}[y_i - f(x_{1i}, x_{2i})]^2 + \lambda \int\int_{\mathbb{R}^2} \|\nabla^2 f(x_1, x_2)\|_F^2 \, dx_1 \, dx_2.$$

The fits from MARS and TPS are incredibly flexible, but can be less interpretable than the

fits from CART.

In recent years, the statistical community has been very interested in formulating predictive models as solutions to convex optimization problems. However, to the best of our knowledge, no proposals have been made for flexible, non-additive, and interpretable modeling via convex optimization. To close this gap, we propose a non-greedy procedure whose fits have a block structure reminiscent of CART. Our proposal, *convex regression with interpretable sharp partitions* (CRISP), is the solution to a convex optimization problem with predictions that are much less variable than those of CART. Also unlike CART, CRISP borrows information across the blocks, and is able to adequately model the data when the mean model is smooth. Thus our method provides a compromise between the interpretability of CART and the flexibility of MARS and TPS. In this chapter, we consider the low-dimensional setting in which there are a small number of covariates of interest ($p \ll n$). We leave an extension to the $p > n$ setting to future work.

CRISP has a number of attractive properties:

- CRISP can accommodate interactions between pairs of covariates in a flexible way. This is useful when the impact of one covariate may depend on the value of another covariate, but there is not strong *a priori* knowledge about the form of the interaction.

- CRISP fits a piecewise constant model, which is easily interpreted by even those with limited statistical background.

- CRISP is formulated as a convex optimization problem. Thus we can solve for the global optimum, and can derive an expression for CRISP's degrees of freedom.

The remainder of this chapter is organized as follows. In Section 3.1, we introduce our method and present an algorithm to implement it. We compare our method to existing approaches using simulated data in Section 3.2. In Section 3.3, we derive some properties of the method. In Section 3.4, we discuss connections between our method and other work. We illustrate our method on a housing price data set in Section 3.5. We consider a modification

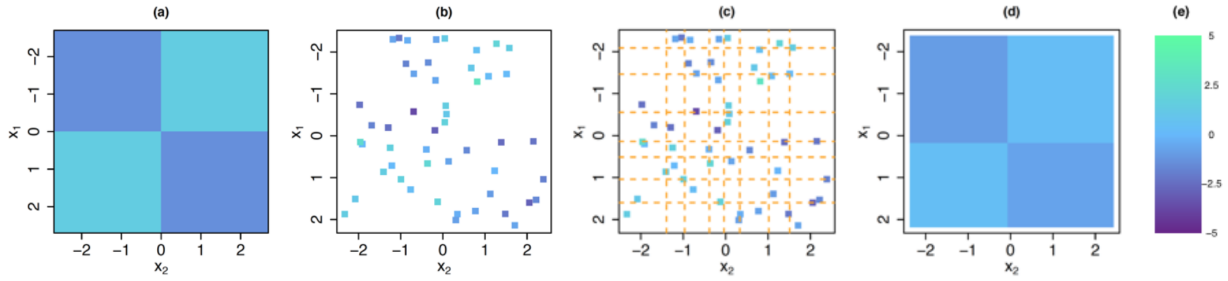


Figure 3.1: In *(a)*, the mean model $f(x_1, x_2)$ used to generate data. In *(b)*, each of the 50 squares represents an observation (x_1, x_2, y) with $y = f(x_1, x_2) + \epsilon$ with $\epsilon \sim N(0, 1)$. In *(c)*, there are $q^2 = 64$ bins of (x_1, x_2) values, whose boundaries coincide with the octiles (- - -) of $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$. In *(d)*, CRISP estimates $f(x_1, x_2)$ to be constant within each bin, and furthermore encourages adjacent bins to take on the same value. When applied to the data in *(b)* with $q = 8$, this leads to an estimated $f(x_1, x_2)$ with four *blocks*. In *(e)*, we show the heat scale legend.

to our proposal in Section 3.6, and close with the discussion in Section 3.7. Proofs are in Appendix B.

3.1 Convex Regression with Interpretable Sharp Partitions

Throughout most of this chapter, for ease of exposition, we focus on the case of $p = 2$ features. An extension to the case of $p > 2$ is given in Section 3.6.

We first present an overview of the CRISP approach. We wish to predict a random variable $y \in \mathbb{R}$ using $x_1, x_2 \in \mathbb{R}$. We assume that $y = f(x_1, x_2) + \epsilon$, where ϵ is a mean-zero error term, and f is an unknown function that we wish to estimate. An example of $f(x_1, x_2)$ is displayed in Figure 3.1(a). Figure 3.1(b) displays a training set of n i.i.d. observations of (x_1, x_2, y). We first partition the feature space into q^2 bins, as shown in Figure 3.1(c) with $q = 8$. The CRISP approach estimates $f(x_1, x_2)$ to be constant within each bin, and further encourages f to take on the same value at adjacent bins; this leads to constant-valued *blocks*. The CRISP output is shown in Figure 3.1(d); there are four estimated blocks. More details about this simulation set-up are provided in Section 3.2.

3.1.1 Notation and Goal of CRISP

We now introduce some new notation, and provide further intuition for CRISP, before presenting the optimization problem for CRISP in Section 3.1.2.

As is shown in Figure 3.1(c), we wish to estimate the mean model $f(x_1, x_2)$ for a $q \times q$ grid of bins, where $f(x_1, x_2)$ is estimated to be constant within each bin. Let $\boldsymbol{M} \in \mathbb{R}^{q \times q}$ denote a mean matrix whose element $M_{(i)(j)}$ contains the mean for pairs of covariate values within a *quantile range* of the observed predictors $\boldsymbol{x}_1, \boldsymbol{x}_2 \in \mathbb{R}^n$. For example, $M_{(1)(2)}$ represents the mean of the observations with x_1 less than the $\frac{1}{q}$-quantile of $\boldsymbol{x}_1$, and x_2 between the $\frac{1}{q}$- and $\frac{2}{q}$-quantiles of $\boldsymbol{x}_2$. In Figure 3.1(c), the corner grid bins correspond to $M_{(1)(1)}$, $M_{(8)(1)}$, $M_{(8)(8)}$, and $M_{(1)(8)}$, starting at the top-left corner of the grid and moving counter-clockwise. In CRISP, our goal is to estimate the $q \times q$ matrix $\boldsymbol{M}$ on the basis of $\boldsymbol{y} \in \mathbb{R}^n$, which contains n noisy observations from various bins of $\boldsymbol{M}$.

In the example shown in Figure 3.1, we partition the feature space into an 8×8 grid (shown in Figure 3.1(c)), which translates to estimating an 8×8 matrix $\boldsymbol{M}$. Therefore, instead of estimating $f(x_1, x_2)$ over the entire joint range of $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$, we need only estimate the 64 elements of $\boldsymbol{M}$. Furthermore, CRISP borrows information across bins of the grid by encouraging pairs of neighboring rows and columns of $\boldsymbol{M}^*$ to be equal, leading to an estimated mean model with a *block structure*. For instance, in Figure 3.1(d), $\boldsymbol{M}$ is estimated to have four *blocks*, or regions of feature space over which $f(x_1, x_2)$ is constant. Consequently, the CRISP solution $\boldsymbol{M}^*$ shown in Figure 3.1(d) only has 4 unique elements, while $\boldsymbol{M}$ is an 8×8 matrix. If we examined the estimate $\boldsymbol{M}^*$, we would see that all pairs of neighboring rows and neighboring columns of $\boldsymbol{M}^*$ are identical, except for one pair of columns and one pair of rows.

While the true mean model in this example has a block structure (as seen in Figure 3.1(a)), we will show in Section 3.2 that CRISP can perform well even when the true mean model is smooth. The data in this example were uniformly distributed in the covariate space. CRISP is most suitable for data applications where observations are distributed throughout

the covariate space. Highly correlated covariates will lead to an insufficient amount of data to estimate the mean model over the entire covariate space.

3.1.2 The Optimization Problem

The CRISP optimization problem balances the trade-off between fitting the data and encouraging a block structure. We estimate $\boldsymbol{M}$ by solving the convex optimization problem

$$\underset{\boldsymbol{M}\in\mathbb{R}^{q\times q}}{\text{minimize}} \quad \frac{1}{2}\sum_{i=1}^{n}\left(y_i - \Omega(\boldsymbol{M}, x_{1i}, x_{2i})\right)^2 + \lambda P(\boldsymbol{M}). \tag{3.1}$$

In (3.1), the function Ω extracts the element of $\boldsymbol{M}$ corresponding to the bin to which the observation (x_{1i}, x_{2i}) belongs. For instance, in Figure 3.1(c), $\Omega(\boldsymbol{M}, 0, -1) = M_{(4)(2)}$. Note that Ω is explicitly defined in Appendix B.1. Furthermore, $\lambda \geq 0$ is a tuning parameter, and the penalty P is defined as

$$P(\boldsymbol{M}) = \sum_{i=1}^{q-1}\left[\left\|\boldsymbol{M}_{i\cdot} - \boldsymbol{M}_{(i+1)\cdot}\right\|_2 + \left\|\boldsymbol{M}_{\cdot i} - \boldsymbol{M}_{\cdot(i+1)}\right\|_2\right], \tag{3.2}$$

where $\boldsymbol{M}_{i\cdot}$ and $\boldsymbol{M}_{\cdot i}$ denote the ith row and column of $\boldsymbol{M}$, respectively. The sum of squared errors in (3.1) encourages the estimate of $\boldsymbol{M}$ to fit the data, while the group lasso penalty [Yuan and Lin, 2006] in (3.2) encourages pairs of neighboring rows (or columns) to be exactly identical. This leads to the formation of constant-valued blocks, which are comprised of multiple bins of the $q \times q$ grid. Appendix B.2 discusses other possible penalties that could be used in (3.1).

We now rewrite (3.1) in a way that will be useful later. We introduce a vectorized form of $\boldsymbol{M}$, which is denoted by $\boldsymbol{m} = \text{vec}(\boldsymbol{M}) = \left((\boldsymbol{M}_{\cdot 1})^T, (\boldsymbol{M}_{\cdot 2})^T, \cdots, (\boldsymbol{M}_{\cdot q})^T\right)^T$ where $\boldsymbol{M}_{\cdot i}$ is the ith column of $\boldsymbol{M}$. The correspondence between $\boldsymbol{M}$ and $\boldsymbol{m}$ is shown in Figure B.1 of Appendix B.1. In what follows, we will switch between using the matrix $\boldsymbol{M}$ and the vectorized $\boldsymbol{m}$. Then (3.1) can be rewritten as

$$\underset{\boldsymbol{m}\in\mathbb{R}^{q^2}}{\text{minimize}} \quad \frac{1}{2}\left\|\boldsymbol{y} - \boldsymbol{Q}\boldsymbol{m}\right\|_2^2 + \lambda\sum_{i=1}^{q-1}\left[\left\|\boldsymbol{R}_i\boldsymbol{m}\right\|_2 + \left\|\boldsymbol{C}_i\boldsymbol{m}\right\|_2\right], \tag{3.3}$$

where each row of $\boldsymbol{Q} \in \mathbb{R}^{n \times q^2}$ contains $q^2 - 1$ elements that equal 0, and a single 1, such that $\boldsymbol{Q}_{i\cdot}\boldsymbol{m} = \Omega(\boldsymbol{M}, x_{1i}, x_{2i})$, where $\boldsymbol{Q}_{i\cdot}$ indicates the ith row of $\boldsymbol{Q}$. (Though $\boldsymbol{Q}$ is a function of $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$, we suppress this to simplify the notation.) In (3.3), $\boldsymbol{R}_i, \boldsymbol{C}_i \in \mathbb{R}^{q \times q^2}$ extract differences between neighboring rows and columns of $\boldsymbol{M}$ (i.e., $\boldsymbol{R}_i\boldsymbol{m} = \boldsymbol{M}_{i\cdot} - \boldsymbol{M}_{(i+1)\cdot}$ and $\boldsymbol{C}_i\boldsymbol{m} = \boldsymbol{M}_{\cdot i} - \boldsymbol{M}_{\cdot(i+1)}$). An example of $\boldsymbol{Q}$ and explicit definitions of $\boldsymbol{Q}$, $\boldsymbol{R}_i$, and $\boldsymbol{C}_i$ are in Appendix B.1. We let $\boldsymbol{A} = \left(\boldsymbol{R}_1^T,\ \cdots,\ \boldsymbol{R}_{q-1}^T,\ \boldsymbol{C}_1^T,\ \cdots,\ \boldsymbol{C}_{q-1}^T\right)^T \in \mathbb{R}^{2q(q-1) \times q^2}$, and then rewrite (3.3) as

$$\underset{\boldsymbol{m} \in \mathbb{R}^{q^2}, \boldsymbol{z} \in \mathbb{R}^{2q(q-1)}}{\text{minimize}} \quad \frac{1}{2}\left\|\boldsymbol{y} - \boldsymbol{Q}\boldsymbol{m}\right\|_2^2 + \lambda \sum_{i=1}^{q-1} \left[\left\|\boldsymbol{z}_{1i}\right\|_2 + \left\|\boldsymbol{z}_{2i}\right\|_2\right] \quad \text{subject to } \boldsymbol{A}\boldsymbol{m} = \boldsymbol{z}, \tag{3.4}$$

where $\boldsymbol{z} = \left((\boldsymbol{z}_{11})^T,\ \cdots,\ (\boldsymbol{z}_{1(q-1)})^T,\ (\boldsymbol{z}_{21})^T,\ \cdots,\ (\boldsymbol{z}_{2(q-1)})^T\right)^T$ with $\boldsymbol{z}_{1i}, \boldsymbol{z}_{2i} \in \mathbb{R}^q$.

While (3.1), (3.3), and (3.4) have the same solution, it is most convenient to derive an algorithm to solve CRISP using the parameterization in (3.4). Throughout this chapter, we will alternate between using the notation $\boldsymbol{M}^*$ and $\boldsymbol{m}^*$, where $\boldsymbol{m}^* = \text{vec}(\boldsymbol{M}^*)$, to represent the CRISP solution to (3.4). The training set predictions for CRISP are given by $\hat{\boldsymbol{y}} = \boldsymbol{Q}\boldsymbol{m}^*$.

3.1.3 An Algorithm for CRISP

We solve for the global optimum of the convex optimization problem (3.4) using the *alternating directions method of multipliers* (ADMM) algorithm [Boyd et al., 2011]. This is summarized in Algorithm 2. Additional details are in Appendix B.3.

In Algorithm 2, the computational bottleneck occurs in Step 2(a). Evaluating the q-banded matrix $\boldsymbol{Q}^T\boldsymbol{Q} + \rho\boldsymbol{A}^T\boldsymbol{A}$ has a one-time cost of $\mathcal{O}(n + q^4)$ operations, and computing its LU factorization requires an additional $\mathcal{O}(q^4)$ operations. Then Step 2(a) can be performed in $\mathcal{O}(q^3)$ operations [Boyd and Vandenberghe, 2004]. Therefore, Algorithm 2 requires an initial step of $\mathcal{O}(n + q^4)$ operations, followed by a per-iteration complexity of $\mathcal{O}(q^3)$.

On a Macbook Pro with a 2.0 GHz Intel Sandy Bridge Core i7 processor, our `Python` implementation of CRISP with $n = q = 50$ takes 20.1 seconds for a sequence of 20 λ values. For $n = q = 100$ and $n = q = 200$, the run times are 84.7 and 383.6 seconds, respectively.

Algorithm 2 — Alternating Directions Method of Multipliers for Equation (3.4)

1. Let $\boldsymbol{u} = \left((\boldsymbol{u}_{11})^T, \ldots, (\boldsymbol{u}_{1(q-1)})^T, (\boldsymbol{u}_{21})^T, \ldots, (\boldsymbol{u}_{2(q-1)})^T\right)^T$ denote the scaled dual variables. Initialize $\boldsymbol{m}^{(0)} := \boldsymbol{0}, \boldsymbol{z}^{(0)} := \boldsymbol{0}$, and $\boldsymbol{u}^{(0)} := \boldsymbol{0}$.

2. For $k = 1, 2, \ldots$, until the primal and dual residuals satisfy a stopping criterion:

   (a) $\boldsymbol{m}^{(k)} := \left[\boldsymbol{Q}^T\boldsymbol{Q} + \rho\boldsymbol{A}^T\boldsymbol{A}\right]^{-1}\left[\boldsymbol{Q}^T\boldsymbol{y} + \rho\boldsymbol{A}^T(\boldsymbol{z}^{(k-1)} - \boldsymbol{u}^{(k-1)})\right]$

   (b) $\boldsymbol{z}_{1i}^{(k)} := (\boldsymbol{R}_i\boldsymbol{m}^{(k)} + \boldsymbol{u}_{1i}^{(k-1)})(1 - \lambda/(\rho\|\boldsymbol{R}_i\boldsymbol{m}^{(k)} + \boldsymbol{u}_{1i}^{(k-1)}\|_2))_+$,
   $\boldsymbol{z}_{2i}^{(k)} := (\boldsymbol{C}_i\boldsymbol{m}^{(k)} + \boldsymbol{u}_{2i}^{(k-1)})(1 - \lambda/(\rho\|\boldsymbol{C}_i\boldsymbol{m}^{(k)} + \boldsymbol{u}_{2i}^{(k-1)}\|_2))_+$ for $i = 1, \ldots, q-1$

   (c) $\boldsymbol{u}^{(k)} := \boldsymbol{u}^{(k-1)} + \boldsymbol{A}\boldsymbol{m}^{(k)} - \boldsymbol{z}^{(k)}$

Increasing n while holding q constant has little effect on the run times; this is consistent with the discussion in the previous paragraph. Thus even for very large n, the computational time is reasonable.

We chose to solve CRISP using an ADMM algorithm, as ADMM works well in related problems. For example, in the context of trend filtering, Ramdas and Tibshirani [forthcoming] found that their ADMM implementation converged more reliably across a variety of tuning parameter values and sample sizes than the primal-dual interior point method of Kim et al. [2009]. In our setting, an interior point algorithm for CRISP involves solving a dense system of equations at each iteration, which has a computational complexity of $\mathcal{O}(q^6)$. Additionally, an interior point method would not recover the exact block structure (any strictly feasible solution would have no zero row or column differences). In contrast, we directly recover the block structure of our estimated mean model from the $\boldsymbol{z}$ variables of our ADMM algorithm. Furthermore, ADMM algorithms typically converge to moderate accuracy within only tens of iterations [Boyd et al., 2011], which is acceptable in our setting.

The value of λ can be chosen using K-fold cross-validation. Alternatively, λ can be selected using approaches based on Akaike's information criterion [AIC; Akaike, 1973] or

Bayesian information criterion [BIC; Schwarz, 1978] using the degrees of freedom estimator proposed in Section 3.3.1. The roles of λ and q in controlling the granularity of the model are further characterized in Sections 3.3.2 and 3.3.3.

3.2 Simulations

In this section, we compare the performance of CRISP to CART, TPS, and competing methods. We consider a variety of mean models, as well as smaller ($n = 100$) and larger ($n = 10,000$) training set sample sizes.

3.2.1 Methods

We generate data with either $n = 100$ or $n = 10,000$, and $p = 2$. We independently sample each element of $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ from a Unif$[-2.5, 2.5]$ distribution, and then take $\boldsymbol{y} = f(\boldsymbol{x}_1, \boldsymbol{x}_2) + \boldsymbol{\epsilon}$, where $\boldsymbol{\epsilon} \sim \text{MVN}(\boldsymbol{0}, \sigma^2 \boldsymbol{I}_n)$ with $\sigma = 1$ for $n = 100$ and $\sigma = 10$ for $n = 10,000$. Note that we use the notation MVN to indicate a multivariate normal distribution.

We consider four mean models for $f(x_1, x_2)$; these are displayed in the top panel of Figure 3.2, and defined in detail in Appendix B.4. In Scenario 1, the mean model is additive in $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$. Scenario 2 is similar to Scenario 1, but the mean model is non-additive. The mean model in Scenario 3 is piecewise constant, with the cut points for $\boldsymbol{x}_2$ depending on $\boldsymbol{x}_1$. Finally, Scenario 4 is a smooth mean model.

For each scenario, we generate 200 data sets and estimate $\boldsymbol{M}$ using CRISP (with $q = 100$) and several competitors: FLAM (implemented with the `R` package `flam` [Petersen, 2014]); CART (implemented with the `R` package `rpart` [Therneau et al., 2014]); TPS (implemented with the `R` package `fields` [Nychka et al., 2014]); a linear model with predictors $\boldsymbol{x}_1$, $\boldsymbol{x}_2$, and their interaction; and an "oracle" linear model based on knowing *a priori* which regions of the mean model take on a constant value.

For each of the four scenarios, we plot mean squared prediction error[1] versus degrees of

[1]Mean squared prediction error is defined as $\frac{1}{q^2}\|\boldsymbol{M} - \boldsymbol{M}^*\|_F^2$, where $\boldsymbol{M} \in \mathbb{R}^{q \times q}$ is the true mean matrix and $\boldsymbol{M}^* \in \mathbb{R}^{q \times q}$ is the estimate from a given method. For methods other than CRISP, $\boldsymbol{M}^*$ was constructed

freedom (a notion that will be discussed extensively in Section 3.3.1). CRISP and FLAM are fit over a sequence of exponentially decreasing λ values, with the degrees of freedom estimated using (3.6) and a result from Petersen et al. [forthcoming], respectively. TPS is fit over a sequence of degrees of freedom. For CART, we vary the number of terminal nodes in the tree, and average the estimator (3.7) over the replicates in order to estimate the degrees of freedom for each number of terminal nodes. Note that the number of degrees of freedom of CART is non-monotonic for small numbers of terminal nodes (as seen in Figure 3.3).

3.2.2 Results for $n = 100$

Results are shown in Figure 3.3. We see that both CRISP and TPS perform reasonably well in terms of prediction error in all scenarios, regardless of the true mean model. FLAM outperforms the other methods in Scenario 1, which is unsurprising as the mean model is truly additive, and FLAM boils down to CRISP with an additivity constraint (Section 3.4.2). However, FLAM performs poorly for mean models with substantial non-additivity (Scenarios 2 and 4). Outside of Scenario 1, CART performs worse than TPS and CRISP. CRISP, TPS, and CART all perform better than a linear model with an interaction in Scenarios 1–3. However, in Scenario 4, the mean model is well-approximated using a linear model. We also fit MARS for all scenarios; however, performance was poor and the results are omitted.

While CRISP and TPS have comparable prediction error, their fits are quite different. In Figure 3.2, we show the estimated mean models for CRISP, TPS, and CART for a single replicate of data in each scenario. CRISP provides fits that reflect the true mean model well, even when the true mean model is smooth. While TPS has low prediction error, the smooth fits from TPS are not easily interpreted and are far from the true mean model in some scenarios. While the fits from CART reflect the mean model reasonably well in Scenarios 1 and 2, the fits from CART in all scenarios are highly variable. CART fits from different replicates of Scenario 4 are shown in Figure 3.4. The average variance of an element of $\boldsymbol{M}^*$

---

using the mean model estimate at the midpoint of each bin of the $q \times q$ grid.

across the 200 replicates for Scenario 4 was 0.843 for CART, compared to 0.0935 for CRISP and 0.0653 for TPS. The variance of CART's fitted values is similarly inflated for the other scenarios. Small perturbations of the data can produce very different qualitative conclusions when examining CART's fits.

3.2.3 Results for $n = 10,000$

We compare CRISP to TPS and CART. Results are in Figures 3.2 and 3.5. Again, CRISP performs well in all scenarios, and the CART fits are much more variable than those of CRISP and TPS. The average variance of an element of $\boldsymbol{M}^*$ across the 200 replicates for Scenario 1 was 0.111 for CART, compared to 0.051 for CRISP and 0.083 for TPS. For Scenario 2, the average variance was 1.42 for CART, compared to 0.056 for CRISP and 0.083 for TPS. For Scenario 3, the average variance was 0.692 for CART, compared to 0.077 for CRISP and 0.129 for TPS. And finally, for Scenario 4, the average variance was 1.89 for CART, compared to 0.096 for CRISP and 0.061 for TPS. Notably, a large sample size is not sufficient for producing stable CART fits, unless the signal-to-noise ratio is suitably large.

3.3 Properties of CRISP

In this section, we provide an unbiased estimator for CRISP's degrees of freedom. We also derive an analytical expression for the range of λ for which the solution to (3.4) takes a constant value, $\boldsymbol{m}^* = \left(\frac{1}{n}\mathbf{1}^T\boldsymbol{y}\right)\mathbf{1}$. Lastly, we discuss the role of q and λ in controlling the granularity of CRISP. Throughout this section, we use $\boldsymbol{A}^+$ to denote the Moore-Penrose pseudoinverse of a matrix $\boldsymbol{A}$.

3.3.1 Degrees of Freedom

Suppose that $\mathrm{Var}(\boldsymbol{y}) = \sigma^2\boldsymbol{I}$, and let $g(\boldsymbol{y}) = \hat{\boldsymbol{y}}$ denote the fit corresponding to some model-fitting procedure g. Then the degrees of freedom of g is defined as $\frac{1}{\sigma^2}\sum_{i=1}^{n}\mathrm{Cov}(y_i, \hat{y}_i)$ [Efron, 1986, Hastie and Tibshirani, 1990].

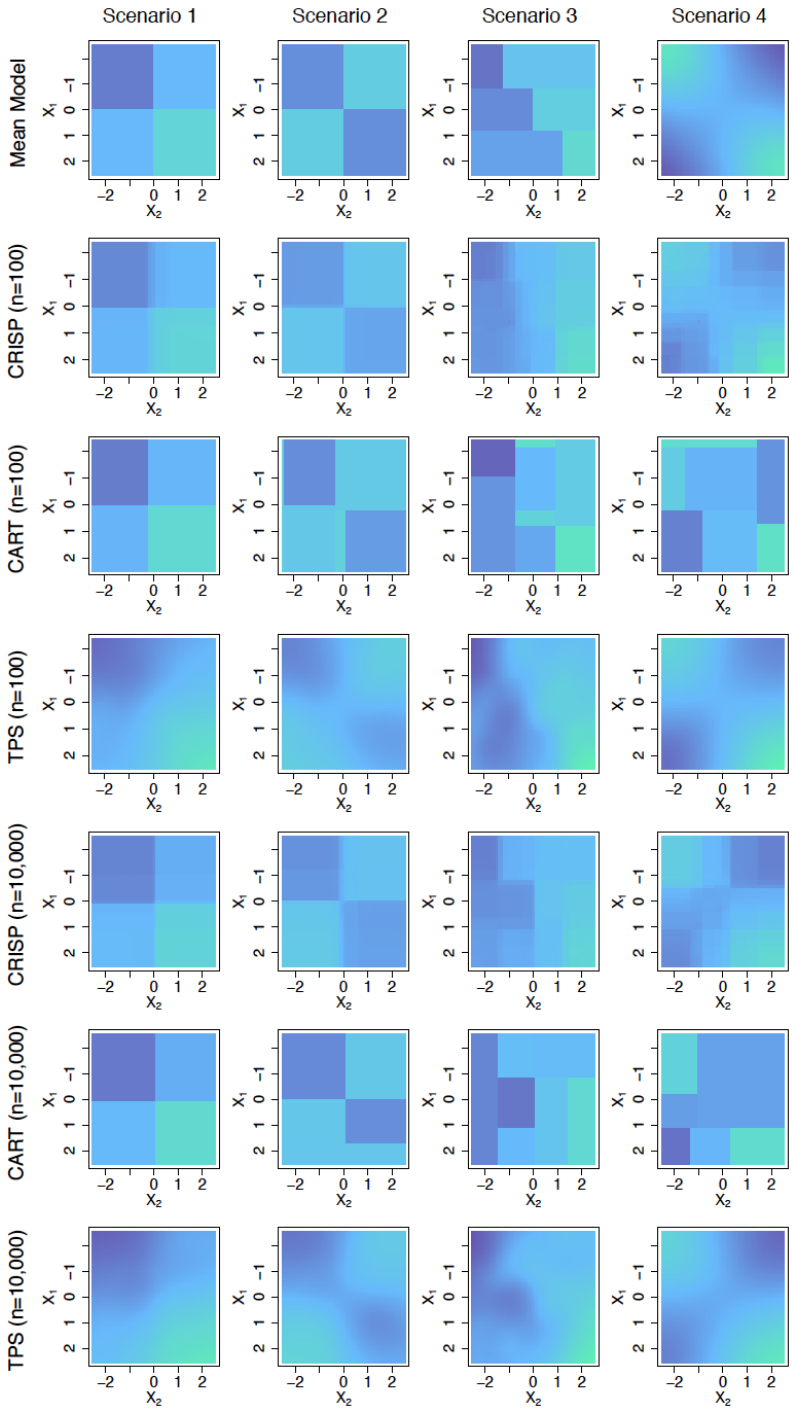


Figure 3.2: The mean models for Scenarios 1–4, as well as estimated mean models from CRISP, CART, and TPS for the simulations considered in Section 3.2. Each fit is from a single replicate of data, with the number of degrees of freedom indicated in Figures 3.3 and 3.5 for $n = 100$ and $n = 10,000$, respectively. The heat scale legend is in Figure 3.1(e).

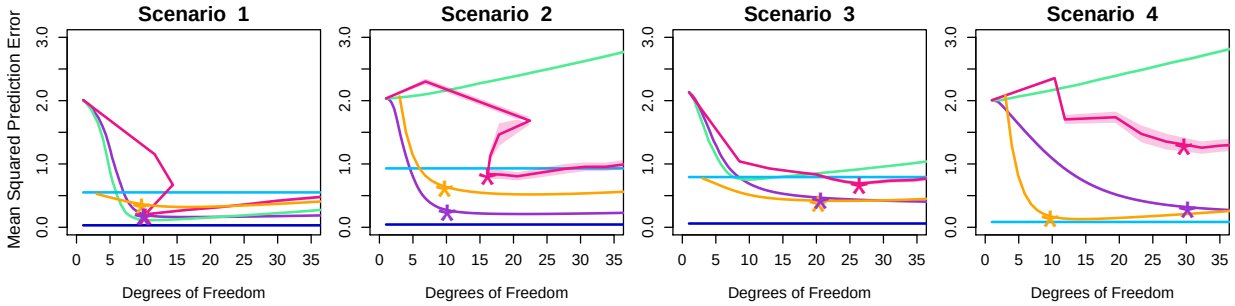


Figure 3.3: Mean squared prediction error, as a function of the degrees of freedom, for the four scenarios considered in the simulations of Section 3.2.2. The methods displayed are CRISP (—), FLAM (—), TPS (—), CART (—), linear model with an interaction (—), and the oracle linear model (—). The oracle linear model is only fit for Scenarios 1–3, for which the mean models have constant regions. Shaded bands (only visible for CART) indicate point-wise 95% confidence intervals over the 200 replicate data sets. The linear models have a fixed number of degrees of freedom, but are shown as horizontal lines. Asterisks indicate the degrees of freedom used for the fits shown in Figure 3.2.

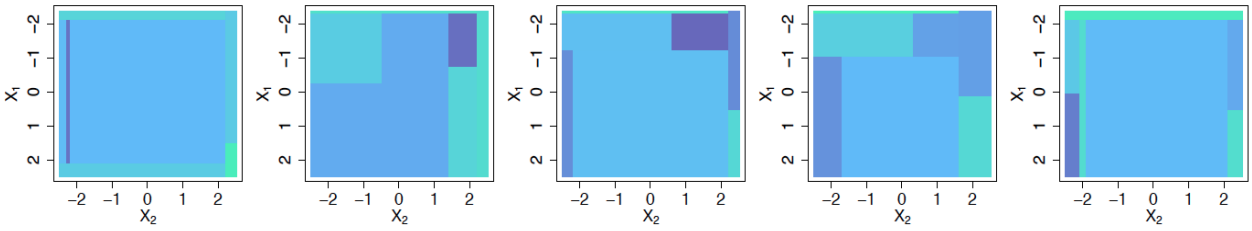


Figure 3.4: Fits for CART in Scenario 4 with $n = 100$ (as also shown in Figure 3.2) corresponding to five additional replicates of data. The heat scale legend is in Figure 3.1(e).

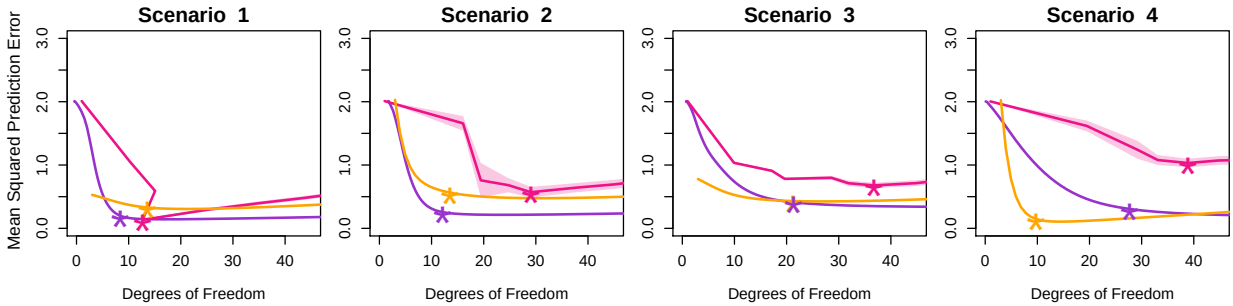


Figure 3.5: Results for $n = 10,000$ for CRISP (—), TPS (—), and CART (—) in the simulations of Section 3.2.3. Details are as given in Figure 3.3.

The concept of degrees of freedom provides a common framework for comparing the complexities of various models; this is particularly useful when the models under consideration are complex or unrelated. Ye [1998] proposed a computationally-burdensome Monte Carlo approach for estimating the degrees of freedom of a model-fitting procedure. In recent years, unbiased estimators for the degrees of freedom have been derived for the lasso and generalized lasso [Tibshirani and Taylor, 2012, Zou et al., 2007], among other methods. These estimators allow us to characterize a model's complexity, and also can be used in order to develop an approach for tuning parameter selection based on Akaike's information criterion [AIC; Akaike, 1973] or Bayesian information criterion [BIC; Schwarz, 1978].

Problem (3.3) is equivalent to the problem

$$\underset{\boldsymbol{m}}{\text{minimize}} \quad \frac{1}{2}\|\boldsymbol{y}-\boldsymbol{Q}\boldsymbol{m}\|_2^2 + \lambda \sum_{i=1}^{q-1} [\|\boldsymbol{R}_i\boldsymbol{m}\|_2 + \|\boldsymbol{C}_i\boldsymbol{m}\|_2] + \frac{\gamma}{2}\|\boldsymbol{m}\|_2^2 \tag{3.5}$$

with $\gamma = 0$. In the rest of this section, we take γ to be a small positive constant, which ensures strong convexity and enforces uniqueness of the solution.

We now introduce some notation. First, we define $\mathcal{C}$, the set of difference matrices corresponding to equal neighboring rows or columns in the solution $\boldsymbol{m}^*$ to (3.5). That is, $\mathcal{C} = \{\boldsymbol{A}_i : \|\boldsymbol{A}_i\boldsymbol{m}^*\|_2 = 0\}$ where $\boldsymbol{A}_1 = \boldsymbol{R}_1, \boldsymbol{A}_2 = \boldsymbol{R}_2, \ldots, \boldsymbol{A}_{q-1} = \boldsymbol{R}_{q-1}, \boldsymbol{A}_q = \boldsymbol{C}_1, \boldsymbol{A}_{q+1} = \boldsymbol{C}_2, \ldots, \boldsymbol{A}_{2q-2} = \boldsymbol{C}_{q-1}$. Then we define $\boldsymbol{A}_*$ to be the submatrix of $\boldsymbol{A}$ obtained by retaining

only the rows of $\boldsymbol{A}$ corresponding to matrices $\boldsymbol{A}_i \in \mathcal{C}$. Note that $\boldsymbol{A}_* \in \mathbb{R}^{q|\mathcal{C}| \times q^2}$. We propose to estimate the degrees of freedom of CRISP as

$$\hat{df}_{CRISP} = \text{Tr}\left[\boldsymbol{Q}\left(\boldsymbol{D} + \lambda \boldsymbol{P} \sum_{i:\boldsymbol{A}_i \notin \mathcal{C}} S_2(\boldsymbol{A}_i, \boldsymbol{m}^*)\boldsymbol{P} + \gamma \boldsymbol{I}\right)^{-1} \boldsymbol{P}\boldsymbol{Q}^T\right], \tag{3.6}$$

where $\boldsymbol{P} = \boldsymbol{I}_{q^2} - \boldsymbol{A}_*^+ \boldsymbol{A}_*$, $S_2(\boldsymbol{A}_i, \boldsymbol{m}^*) = \frac{\boldsymbol{A}_i^T \boldsymbol{A}_i}{\|\boldsymbol{A}_i \boldsymbol{m}^*\|_2} - \frac{\boldsymbol{A}_i^T \boldsymbol{A}_i \boldsymbol{m}^* \boldsymbol{m}^{*T} \boldsymbol{A}_i^T \boldsymbol{A}_i}{\|\boldsymbol{A}_i \boldsymbol{m}^*\|_2^3}$, and $\boldsymbol{Q}$ was defined in (3.3). Recall that $\boldsymbol{M}^*$ will tend to contain row-column blocks of constant value, as shown in Figure 3.1(d). We define $\boldsymbol{D} = \text{diag}\left(h(m_1^*), \cdots, h(m_{q^2}^*)\right)$, where $h(m_i^*)$ is the ratio of the number of observations in the block of $\boldsymbol{M}^*$ that contains m_i^* to the number of elements of $\boldsymbol{M}^*$ in the block of $\boldsymbol{M}^*$ that contains m_i^*. We use the notation MVN to indicate a multivariate normal distribution.

Proposition 3.3.1 *Assume $\boldsymbol{y} \sim MVN(\boldsymbol{\mu}, \sigma^2 \boldsymbol{I})$. Then $\hat{df}_{CRISP}$ is an unbiased estimator of the degrees of freedom of CRISP.*

The following corollary indicates that the estimator (3.6) simplifies substantially when the CRISP solution takes a particular form.

Corollary 3.3.2 *Assume $\boldsymbol{y} \sim MVN(\boldsymbol{\mu}, \sigma^2 \boldsymbol{I})$. If either all rows or all columns of $\boldsymbol{M}^*$ are equal, then the total number of blocks of $\boldsymbol{M}^*$ is an unbiased estimator of the degrees of freedom.*

In 100 replicate data sets with $y_i \sim N(\mu_i, \sigma^2)$, we compare the mean of (3.6) to the mean of

$$\frac{1}{\sigma^2} \sum_{i=1}^{n} (\hat{y}_i - \mu_i)(y_i - \mu_i), \tag{3.7}$$

which provides a Monte Carlo estimate of $\frac{1}{\sigma^2} \sum_{i=1}^{n} \text{Cov}(y_i, \hat{y}_i)$, the true degrees of freedom of CRISP. The results in Figure 3.6(a) empirically validate Proposition 3.3.1, showing that (3.6) is an unbiased estimator of CRISP's degrees of freedom. Note that the proofs of Proposition 3.3.1 and Corollary 3.3.2 can be found in Appendices B.5 and B.6, respectively.

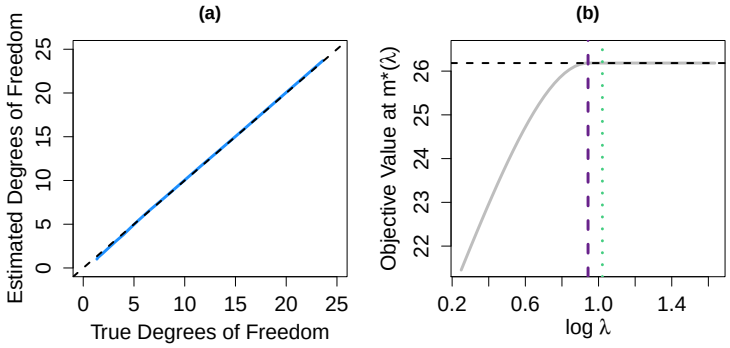


Figure 3.6: In *(a)*, we compare the degrees of freedom calculated using our estimator (3.7) (y-axis) from Section 3.3.1 to the unbiased, Monte Carlo estimator (3.6) (x-axis). Varying λ gives the solid line, and the dashed line indicates $y = x$. In *(b)*, we plot the value of the objective of (3.4) at $\boldsymbol{m}^*(\lambda)$, the minimizer of (3.4) at λ, for a replicate of data as λ varies. We compare two ways of finding a λ large enough such that $\boldsymbol{m}^*(\lambda) = \left(\frac{1}{n}\mathbf{1}^T\boldsymbol{y}\right)\mathbf{1}$, which results in the objective shown as - - -. We take $\lambda = \max_{1\le i\le q-1}\{\|\boldsymbol{d}_{1i}\|_2, \|\boldsymbol{d}_{2i}\|_2\}$ with either $\boldsymbol{d}$ being the solution to (3.8) (— —) or $\boldsymbol{d} = (\boldsymbol{A}^T)^+\boldsymbol{Q}^T\left(\boldsymbol{y} - \left(\frac{1}{n}\mathbf{1}^T\boldsymbol{y}\right)\mathbf{1}\right)$ (· · · ·). The former (— —) matches the result of Lemma 3.3.3 in Section 3.3.2.

3.3.2 Range of λ that Yields a Constant Solution

CRISP has a single tuning parameter λ, which we typically will select via cross-validation or a related approach. Here, we derive the minimum value of λ such that $\boldsymbol{m}^* = \left(\frac{1}{n}\mathbf{1}^T\boldsymbol{y}\right)\mathbf{1}$, corresponding to a fit in which all elements of $\boldsymbol{m}^*$ are equal.

Lemma 3.3.3 *The solution to* (3.4) *is constant (i.e.,* $\boldsymbol{m}^* = \left(\frac{1}{n}\mathbf{1}^T\boldsymbol{y}\right)\mathbf{1}$*) if and only if*

$$\lambda \ge \max_{1\le i\le q-1}\{\|\boldsymbol{d}^*_{1i}\|_2, \|\boldsymbol{d}^*_{2i}\|_2\},$$

where $\boldsymbol{d}^* = (\boldsymbol{d}^{*T}_{11} \cdots \boldsymbol{d}^{*T}_{1(q-1)}\ \boldsymbol{d}^{*T}_{21} \ldots \boldsymbol{d}^{*T}_{2(q-1)})^T$ *is the solution to*

$$\underset{\boldsymbol{d}}{\text{minimize}} \quad \max_{1\le i\le q-1}\{\|\boldsymbol{d}_{1i}\|_2, \|\boldsymbol{d}_{2i}\|_2\} \quad \textit{subject to} \quad \boldsymbol{Q}^T\left(\boldsymbol{y} - \left(\frac{1}{n}\mathbf{1}^T\boldsymbol{y}\right)\mathbf{1}\right) = \boldsymbol{A}^T\boldsymbol{d}. \tag{3.8}$$

Recall that the matrix $\boldsymbol{Q}$ was defined in (3.3). Taking $\lambda = \max_{1\le i\le q-1}\left\{\|\tilde{\boldsymbol{d}}_{1i}\|_2, \|\tilde{\boldsymbol{d}}_{2i}\|_2\right\}$ for any feasible vector $\tilde{\boldsymbol{d}}$ for (3.8) will give a value of λ sufficiently large so $\boldsymbol{m}^*$ is constant. For

example, we can choose $\tilde{\boldsymbol{d}} = (\boldsymbol{A}^T)^+ \boldsymbol{Q}^T \left(\boldsymbol{y} - \left(\frac{1}{n}\mathbf{1}^T\boldsymbol{y}\right)\mathbf{1}\right)$. However, choosing λ in accordance with Lemma 3.3.3 will give the minimum value of λ such that $\boldsymbol{m}^* = \left(\frac{1}{n}\mathbf{1}^T\boldsymbol{y}\right)\mathbf{1}$. The optimization problem (3.8) can be solved using a standard convex solver, such as SDPT3 via `CVX` in `MATLAB` [Grant and Boyd, 2008, 2014]. An illustration of Lemma 3.3.3 is provided in Figure 3.6(b).

3.3.3 Controlling the Granularity of CRISP

Both q and λ control the granularity of the final CRISP model: q controls the size of the grid used to construct $\boldsymbol{M}$, and λ controls the number of blocks in the final fitted CRISP model. For a range of very small λ values, there will be q^2 blocks; for larger λ values, the CRISP solution will have a smaller number of blocks.

Given that q and λ both influence the number of blocks in the final fitted CRISP model, one might wonder whether it is necessary to have both q and λ. We illustrate the value of both q and λ through some simple examples.

Choice of q

In principle, q may be chosen to equal n. This means that each bin of the $q \times q$ grid would contain at most one observation. However, when n is large, choosing $q = n$ can lead to excessive computational time, memory burden, and variance in the fit. Instead, we aim to choose q to be large enough to allow for adequate granularity, but not excessively large. What constitutes adequate granularity will depend on the context of the problem.

In our analyses, we choose to treat q as a fixed parameter that is chosen prior to fitting CRISP. However, if desired, q could be chosen by K-fold cross validation.

Choice of λ

To illustrate the role of λ, consider taking $\lambda = 0$ in (3.3), and treating q as a tuning parameter rather than a fixed value. When $\lambda = 0$, (3.3) contains only a sum of squared errors term,

so the estimate within each bin is the mean value of the observations in that bin. For bins without any observations, we estimate the corresponding element of $\boldsymbol{M}$ to be the overall mean of $\boldsymbol{y}$.

For the mean models shown in Figure 3.2, we compare CRISP to (3.3) with $\lambda = 0$ and q chosen adaptively. We focus on the general findings here, but detailed results are given in Appendix B.8. When the true mean model is piecewise constant with boundaries that are well-approximated by a grid of bins (as in Scenarios 1–3), CRISP and (3.3) with $\lambda = 0$ and variable q perform similarly. However, CRISP is clearly superior at estimating the smooth mean model of Scenario 4 (Figure B.3), as it is able to borrow information across bins, instead of simply fitting the mean of observations within each bin. CRISP also allows the granularity of the fitted model to vary adaptively over the covariate space, as shown in Figure B.4(a) of Appendix B.8. The blocks of this mean model perfectly align with a grid that has $q = 3$, but the mean model only has 4 blocks. While (3.3) with $\lambda = 0$ and $q = 3$ fits 9 blocks, CRISP correctly identifies 4 blocks (Figures B.4(b) and B.4(c) of Appendix B.8).

3.4 Connections to Other Methods

In this section, we establish connections between CRISP and two previous proposals.

3.4.1 Connection to One-Dimensional Fused Lasso

Suppose that for a given value of λ, the CRISP fit involves only one covariate: that is, $\boldsymbol{M}^* = \tilde{\boldsymbol{m}}\mathbf{1}_q^T$ or $\boldsymbol{M}^* = \mathbf{1}_q\tilde{\boldsymbol{m}}^T$ for some $\tilde{\boldsymbol{m}} \in \mathbb{R}^q$. We will now show that in this setting, the CRISP solution can be recovered by solving a one-dimensional fused lasso problem [Tibshirani et al., 2005].

Before presenting Lemma 3.4.1, we introduce some notation. Define $\boldsymbol{D} = [\boldsymbol{I}_{(q-1)\times(q-1)}\ \mathbf{0}_{(q-1)\times 1}] - [\mathbf{0}_{(q-1)\times 1}\ \boldsymbol{I}_{(q-1)\times(q-1)}]$ to be the first difference matrix. Define $\tilde{\boldsymbol{y}} \in \mathbb{R}^q$ such that $\tilde{y}_i$ is the mean outcome value of the observations in the ith row of the $q \times q$ grid used to construct $\boldsymbol{M}$. Let n_i denote the number of observations in the ith row of the $q \times q$ grid used to construct $\boldsymbol{M}$. Define $\boldsymbol{W} \in \mathbb{R}^{q\times q}$ to be the diagonal matrix with entries $\sqrt{n_1}, \sqrt{n_2}, \ldots, \sqrt{n_q}$.

Lemma 3.4.1 *Suppose that, for some value of λ, the CRISP solution is of the form $\boldsymbol{M}^* = \tilde{\boldsymbol{m}}\mathbf{1}_q^T$ for some $\tilde{\boldsymbol{m}} \in \mathbb{R}^q$. Then $\tilde{\boldsymbol{m}}$ is the solution to the problem*

$$\underset{\tilde{\boldsymbol{m}} \in \mathbb{R}^q}{\text{minimize}} \quad \frac{1}{2} \left\| \boldsymbol{W}(\tilde{\boldsymbol{y}} - \tilde{\boldsymbol{m}}) \right\|_2^2 + \lambda\sqrt{q} \left\| \boldsymbol{D}\tilde{\boldsymbol{m}} \right\|_1 . \tag{3.9}$$

If instead $\boldsymbol{M}^* = \mathbf{1}_q\tilde{\boldsymbol{m}}^T$, then a result similar to Lemma 3.4.1 holds, with modifications to the definitions of $\boldsymbol{W}$ and $\tilde{\boldsymbol{y}}$.

Equation 3.9 is a weighted fused lasso problem with response vector $\tilde{\boldsymbol{y}}$ and weights $\sqrt{n_1}, \sqrt{n_2}, \ldots, \sqrt{n_q}$. When $q = n$, (3.9) simplifies to a standard one-dimensional fused lasso problem.

Corollary 3.4.2 *If $q = n$ and $\boldsymbol{M}^* = \tilde{\boldsymbol{m}}\mathbf{1}_n^T$, then $\tilde{\boldsymbol{m}}$ is the solution to the one-dimensional fused lasso problem*

$$\underset{\tilde{\boldsymbol{m}} \in \mathbb{R}^n}{\text{minimize}} \quad \frac{1}{2} \left\| \boldsymbol{P}\boldsymbol{y} - \tilde{\boldsymbol{m}} \right\|_2^2 + \lambda\sqrt{n} \left\| \boldsymbol{D}\tilde{\boldsymbol{m}} \right\|_1 , \tag{3.10}$$

where $\boldsymbol{P}$ is the permutation matrix that orders the elements of $\boldsymbol{x}_1$ from least to greatest.

If instead $\boldsymbol{M}^* = \mathbf{1}_n\tilde{\boldsymbol{m}}^T$, then Corollary 3.4.2 holds with $\boldsymbol{P}$ defined to be the permutation matrix that orders the elements of $\boldsymbol{x}_2$ from least to greatest.

3.4.2 Connection to Fused Lasso Additive Model

In this subsection, we will establish that CRISP is a generalization of the fused lasso additive model (FLAM) proposal of Petersen et al. [forthcoming]. FLAM fits an additive model in which each covariate's fit is estimated to be piecewise constant with adaptively-chosen knots.

For simplicity, assume that $q = n$. Consider a modification of CRISP in which we impose additivity on the mean matrix $\boldsymbol{M}$. That is, we assume $f(x_1, x_2) = \theta_0 + f_1(x_1) + f_2(x_2)$, where θ_0 is an overall mean, and f_1 and f_2 are mean-zero over the training observations. We introduce the n-vectors $\boldsymbol{\theta}_1$ and $\boldsymbol{\theta}_2$, where $f_1(x_{i1}) = \theta_{1i}$ and $f_2(x_{i2}) = \theta_{2i}$ for all $i = 1, \ldots, n$. Thus the additivity constraint for the (i, j) element of $\boldsymbol{M}$, $M_{(i)(j)}$, can be expressed as

$$M_{(i)(j)} = \theta_0 + \theta_{1i} + \theta_{2j} \text{ for } i = 1, \ldots, n;\ j = 1, \ldots, n \quad \text{with} \quad \mathbf{1}^T\boldsymbol{\theta}_1 = \mathbf{1}^T\boldsymbol{\theta}_2 = 0. \tag{3.11}$$

Lemma 3.4.3 *CRISP* (3.1)–(3.2) *with* $q = n$ *and with the additional additivity constraint* (3.11) *is equivalent to FLAM with* $p = 2$*, which is the solution to the optimization problem*

$$\begin{aligned} &\underset{\theta_0 \in \mathbb{R}, \boldsymbol{\theta_1}, \boldsymbol{\theta_2} \in \mathbb{R}^n}{\text{minimize}} \quad \frac{1}{2} \left\| \boldsymbol{y} - (\theta_0 \mathbf{1} + \boldsymbol{\theta_1} + \boldsymbol{\theta_2}) \right\|_2^2 + \lambda \left(\left\| \boldsymbol{D P}_1 \boldsymbol{\theta}_1 \right\|_1 + \left\| \boldsymbol{D P}_2 \boldsymbol{\theta}_2 \right\|_1 \right) \\ &\textit{subject to } \mathbf{1}^T \boldsymbol{\theta}_1 = \mathbf{1}^T \boldsymbol{\theta}_2 = \mathbf{0}, \end{aligned} \tag{3.12}$$

where $\lambda \geq 0$ *is a tuning parameter,* $\boldsymbol{P}_j$ *is the permutation matrix that orders the elements of* $\boldsymbol{x}_j$ *from least to greatest, and* $\boldsymbol{D} = [\boldsymbol{I}_{(n-1)\times(n-1)} \ \mathbf{0}_{(n-1)\times 1}] - [\mathbf{0}_{(n-1)\times 1} \ \boldsymbol{I}_{(n-1)\times(n-1)}]$ *is the first difference matrix.*

The proof of Lemma 3.4.3 follows from algebraic manipulation.

CRISP (3.1)–(3.2) with the additivity constraint (3.11) is also equivalent to FLAM when the ℓ_2 norms in the penalty (3.2) are changed to ℓ_1 or ℓ_∞ norms. These alternative penalties are discussed further in Appendix B.2.

Lemma 3.4.3 can be generalized in order to establish that CRISP with $q < n$ is equivalent to a version of FLAM that re-weights the loss function in (3.12) appropriately.

3.5 Data Application

We consider predicting median house value on the basis of median income and average occupancy, measured for 20,640 neighborhoods in California. The data set was originally considered in Pace and Barry [1997] and is publicly available from the Carnegie Mellon StatLib data repository (`lib.stat.cmu.edu`).

For this analysis, we focus on predicting median house value for the central area of the covariate space. In particular, we filter the neighborhoods to select those with median incomes and average occupancies that both fall within the central 95% of the covariate distribution, which results in 18,662 neighborhoods to be analyzed. Further details are provided in Appendix B.9. To illustrate the impact that the size of the data set may have on the preferred analysis approach, we consider five different training set sizes: 100, 500, 1000, 5000, and 11,198 (which corresponds to 60% of the observations). We use the observations not selected for the training set as the test set. For each training set size, we consider 10

different data samples. We compare the performance of CRISP (with $q = 100$) to CART and TPS.

Figure 3.7 shows that income is positively associated with house value. Occupancy is not strongly associated with house value in low-income neighborhoods. However, among neighborhoods with median incomes exceeding around \$50,000, neighborhoods with mostly single or double occupancy tend to have more expensive homes than those with higher occupancies and the same income. This is perhaps because single people and couples without children have more disposable income to spend on housing than families at the same income level.

In Figure 3.7, we show estimated mean models from CRISP for two different values of λ. The larger value of λ has slightly worse prediction performance, but has a simple block structure reminiscent of CART. The smaller value of λ gives better prediction performance with a more complex fit structure that resembles the fits from TPS. This illustrates how CRISP's tuning parameter, λ, balances the trade-off between interpretability and prediction performance.

While the fit from CART in Figure 3.7 is quite interpretable, CART gives highly-variable fits across different splits of the data. This is illustrated in Figure 3.8. The average variance of predictions from CART across the 10 splits of data is more than three times that of CRISP and TPS. For larger training sets, the variance decreases, though the variability of the CART predictions remains much larger than that of CRISP and TPS. In Figure 3.8, we also see that CART's performance in terms of test set mean squared error (MSE) is worse than CRISP and TPS, but becomes increasingly similar with larger sample sizes. For example, in Figure 3.9, we show the results for the largest training set sample considered ($n = 11,198$). We see that all three methods perform very similarly in terms of test set MSE, and provide qualitatively similar estimated mean models. As the available sample size increases, the differences between CRISP, TPS, and CART in terms of prediction performance and interpretability of fits become less pronounced.

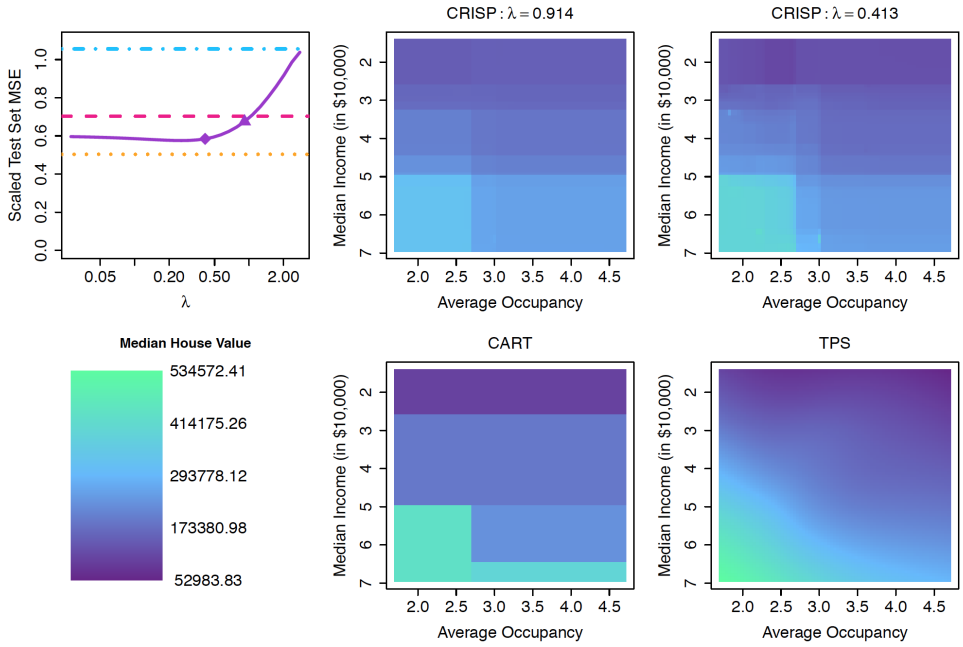


Figure 3.7: We consider predicting median house value on the basis of median income and average occupancy using a training set of size $n = 100$, as considered in Section 3.5. We plot the average value for 10 data samples of test set MSE divided by the variance of the training set outcome. We plot this scaled test set MSE versus λ for CRISP (▬), and show the minimum scaled test set MSE achieved by CART (▬ ▬), TPS (▬ ▬), and an intercept-only model (▬ ▬). Estimated mean models for CRISP are shown for a larger value of λ (indicated by ▲) and a smaller value (indicated by ◆). The estimated mean models shown for CART and TPS correspond to the tuning parameter with the minimum test set MSE. The heat scale legend for the median house value is shown.

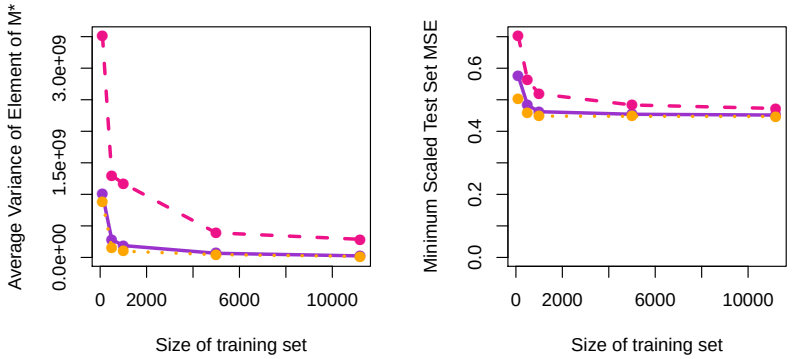


Figure 3.8: We plot the average variance of predictions and the minimum scaled test set MSE (as defined in Figure 3.7) as a function of training set sample size for CRISP (—), CART (— —), and TPS (- - -) applied to the housing data considered in Section 3.5.

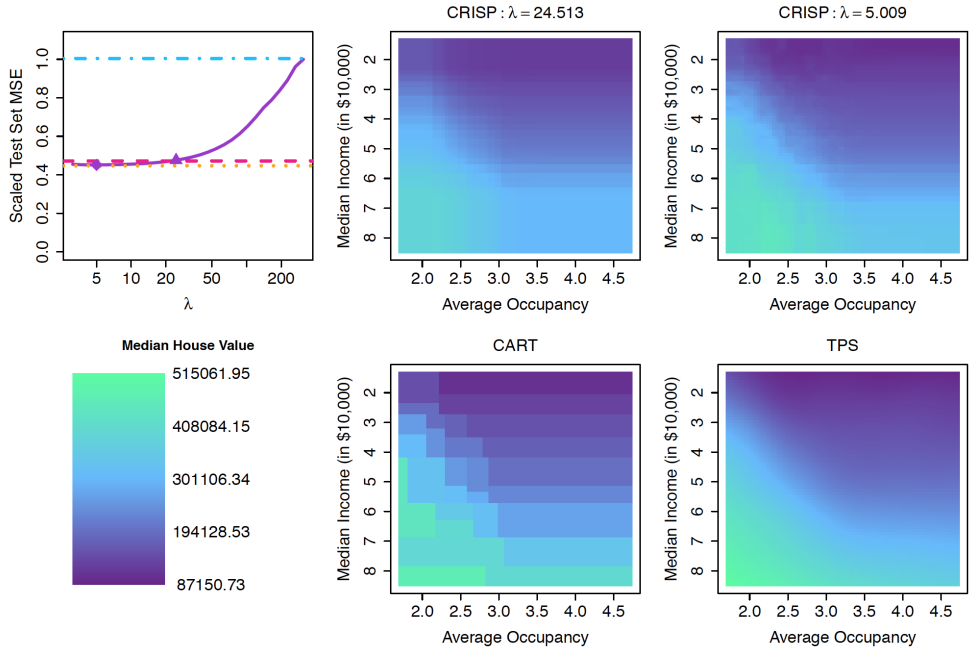


Figure 3.9: Results using median income and average occupancy as predictors of median house value using a training set of size $n = 11,198$, as considered in Section 3.5. Details are as in Figure 3.7.

3.6 Extension to $p > 2$

We have assumed thus far that $p = 2$. In this case, the estimated mean model for the entire covariate space can be summarized in a single plot, as in Figure 3.2.

We extend CRISP to the setting of $p > 2$ by constructing an *additive model of bivariate fits*. That is, we estimate the fit for each of the $\frac{p(p-1)}{2}$ *pairs* of features, giving a bivariate fit for each pair of covariates like those obtained in the setting of $p = 2$ and shown in Figure 3.2. We assume that the mean model is additive in these fits. We restrict the model to pairwise interactions between covariates for a couple of reasons. First, only considering pairwise interactions increases interpretability and reduces model complexity. Our model fit with pairwise interactions can be summarized using $\frac{p(p-1)}{2}$ plots, like those shown in Figure 3.2. There is no analogous way to easily summarize the model if we were to include higher-order interactions. Second, considering higher-order interactions would cause our model to suffer from the *curse of dimensionality*. That is, as the number of covariates increases, the data in any region of the p-dimensional space will become sparser and sparser: there would be an insufficient density of data throughout the covariate space to reasonably estimate a mean model with higher-order interactions.

We now present the details of our proposal for CRISP with $p > 2$. We consider interactions between each pair of features, $\{(j, j') : 1 \leq j < j' \leq p\}$. For ease of notation, we refer to the elements of this set using the index $k \in (1, \ldots, K)$ where $K = \frac{p(p-1)}{2}$. Recall that for $p = 2$, the mean model for CRISP is $\mathrm{E}[\boldsymbol{y} \mid \boldsymbol{x}_1, \boldsymbol{x}_2] = \boldsymbol{Q}\boldsymbol{m}$, where $\boldsymbol{m} \in \mathbb{R}^{q^2}$ is the vectorized mean matrix and $\boldsymbol{Q}$ selects the elements of $\boldsymbol{m}$ corresponding to the covariate bins of the elements of $\boldsymbol{y}$. Recall that $\boldsymbol{Q}$ is a function of $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$, though we suppress this to simplify the notation. For $p > 2$, we consider the mean model

$$\mathrm{E}[\boldsymbol{y} \mid \boldsymbol{x}_1, \ldots, \boldsymbol{x}_p] = m_0\mathbf{1} + \sum_{k=1}^{K} \boldsymbol{Q}_k\boldsymbol{m}_k,$$

where $m_0 \in \mathbb{R}$ is an intercept, $\boldsymbol{m}_k \in \mathbb{R}^{q^2}$ is the vectorized mean matrix for the pair of features indexed by k, and $\boldsymbol{Q}_k \in \mathbb{R}^{n \times q^2}$ selects the elements of $\boldsymbol{m}_k$ corresponding to the covariate

bins for the pair of covariates indexed by k. We include the intercept $m_0 \in \mathbb{R}$ in our model, and assume that $\boldsymbol{m}_1, \ldots, \boldsymbol{m}_K$ are mean-zero, to ensure identifiability.

When $p > 2$, we extend the CRISP optimization problem (3.4) as follows:

$$\begin{aligned} &\underset{m_0, \boldsymbol{m}_k, \boldsymbol{z}_k : k=1,\ldots,K}{\text{minimize}} \quad \frac{1}{2} \left\| \boldsymbol{y} - \left(m_0 \mathbf{1} + \sum_{k=1}^{K} \boldsymbol{Q}_k \boldsymbol{m}_k \right) \right\|_2^2 + \lambda \sum_{k=1}^{K} \sum_{i=1}^{q-1} \left[\|\boldsymbol{z}_{k,1i}\|_2 + \|\boldsymbol{z}_{k,2i}\|_2 \right] \\ &\text{subject to } \boldsymbol{A}\boldsymbol{m}_k = \boldsymbol{z}_k, \mathbf{1}^T \boldsymbol{m}_k = 0, \end{aligned} \tag{3.13}$$

where $\boldsymbol{A}$ is as defined in Section 3.1.2. Thus $\hat{\boldsymbol{y}} = m_0^* \mathbf{1} + \sum_{k=1}^{K} \boldsymbol{Q}_k \boldsymbol{m}_k^*$, where $(m_0^*, \boldsymbol{m}_1^*, \ldots, \boldsymbol{m}_K^*)$ is the solution to (3.13).

Problem (3.13) can be solved using block coordinate descent [Tseng, 2001], which gives Algorithm 3. We iterate through the pairs of covariates, and perform a partial minimization (using Algorithm 2) for each $\boldsymbol{m}_k$, while keeping the others fixed. Using an argument similar to that in Section 3.1.3, the computational complexity of Algorithm 3 is $\mathcal{O}(K(n+q^4))$ for an initial step and $\mathcal{O}(q^3)$ for each iteration of Step 2(b) of Algorithm 3. In practice, the number of iterations needed to achieve convergence in Step 2(b) of Algorithm 3 is relatively small.

We present a block coordinate descent algorithm, since it is a natural extension of Algorithm 2 to the $p > 2$ setting. However, CRISP with $p \gg 2$ can alternatively be fit using generalized gradient descent, which allows the updates for each bivariate fit to be run in parallel on a cluster.

3.7 Discussion

We have presented CRISP, a method for fitting interpretable, flexible, and non-additive predictive models. CRISP fits have an easily-interpreted block structure, which is somewhat reminiscent of the fits from CART. But the fits from CRISP result from a non-greedy procedure, and are much less variable than those of CART. In our numerical studies, the prediction performance of CRISP is similar to TPS, and in many cases CRISP provides a simpler and more interpretable fit.

Algorithm 3 — Block Coordinate Descent for CRISP with $p > 2$ (Equation (3.13))

1. Initialize $m_0^* = 0$ and $\boldsymbol{m}_k^* = \mathbf{0}$ for all $k = 1, \ldots, K$.

2. For $k = 1, \ldots, K, 1, \ldots, K, \ldots$, until convergence of the objective of (3.13):

   (a) Compute the residual $\boldsymbol{r}_k = \boldsymbol{y} - \left(m_0^* \mathbf{1} + \sum_{k' \neq k} \boldsymbol{Q}_{k'} \boldsymbol{m}_{k'}^*\right)$.

   (b) Using Algorithm 2, solve

   $$\underset{\boldsymbol{m}_k, \boldsymbol{z}_k}{\text{minimize}} \quad \frac{1}{2} \|\boldsymbol{r}_k - \boldsymbol{Q}_k \boldsymbol{m}_k\|_2^2 + \lambda \sum_{i=1}^{q-1} \left[\|\boldsymbol{z}_{k,1i}\|_2 + \|\boldsymbol{z}_{k,2i}\|_2\right] \quad \text{subject to } \boldsymbol{A}\boldsymbol{m}_k = \boldsymbol{z}_k.$$

   Let $\boldsymbol{m}_k^*$ denote the solution.

   (c) Compute the intercept, $m_0^* \leftarrow m_0^* + \text{mean}(\boldsymbol{m}_k^*)$, and center, $\boldsymbol{m}_k^* \leftarrow \boldsymbol{m}_k^* - \text{mean}(\boldsymbol{m}_k^*)$.

Future work could consider an alternative penalization scheme. Recall that CRISP first divides the covariate space into a $q \times q$ grid of bins. Our proposal only uses the information about the bin into which each of the n observations falls, which is used to construct $\boldsymbol{Q}$ in (3.4). Thus CRISP only makes use of the rankings of the observations for each covariate, rather than the actual values of the covariates. A modification to (3.4) could allow us to more heavily penalize the differences between pairs of neighboring rows or columns corresponding to observations with similar values in a given covariate. This modification is not very important when the covariate pairs are distributed uniformly over the covariate space, as in our simulation study in Section 3.2.

In this chapter, we have only considered the setting of $p \ll n$. An extension of CRISP to larger p is left to future work.

Chapter 4

IDENTIFYING NEURONS FROM CALCIUM IMAGING DATA

The field of neuroscience is undergoing a transformation: new technologies are making it possible to image activity in large populations of neurons at cellular resolution in behaving animals [Ahrens et al., 2013, Dombeck et al., 2007, Huber et al., 2012, Prevedel et al., 2014]. The resulting data sets promise to provide unprecedented insight into neural activity. However, they bring with them both statistical and computational challenges.

To begin, we briefly describe the science underlying *calcium imaging* data. When a neuron fires, voltage-gated calcium channels in the axon terminal open, and calcium floods the cell. Therefore, intracellular calcium concentration is a surrogate marker for the spiking activity of neurons [Grienberger and Konnerth, 2012]. In recent years, genetically encoded calcium indicators have been developed [Chen et al., 2013, Looger and Griesbeck, 2012, Rochefort et al., 2008]. These indicators bind to intracellular calcium molecules and fluoresce. Thus, the location and timing of neurons firing can be seen through a sequence of two-dimensional images taken over time, typically using two-photon microscopy [Helmchen and Denk, 2005, Svoboda and Yasuda, 2006].

A typical calcium imaging video consists of a 500×500 pixels frame over 1 hour, sampled at 15-30 Hz. A given pixel in a given frame is continuous-valued, with larger values representing higher fluorescent intensities due to greater calcium concentrations. An example frame from a calcium imaging video is shown in Figure 4.1(a).

On the basis of a calcium imaging video, two goals are typically of interest:

- *Neuron identification:* The goal is to assign pixels of the image frame to neurons. Due to the thickness of the brain slice captured by the imaging technology, neurons can overlap in the two-dimensional image. This means that a single pixel can be assigned

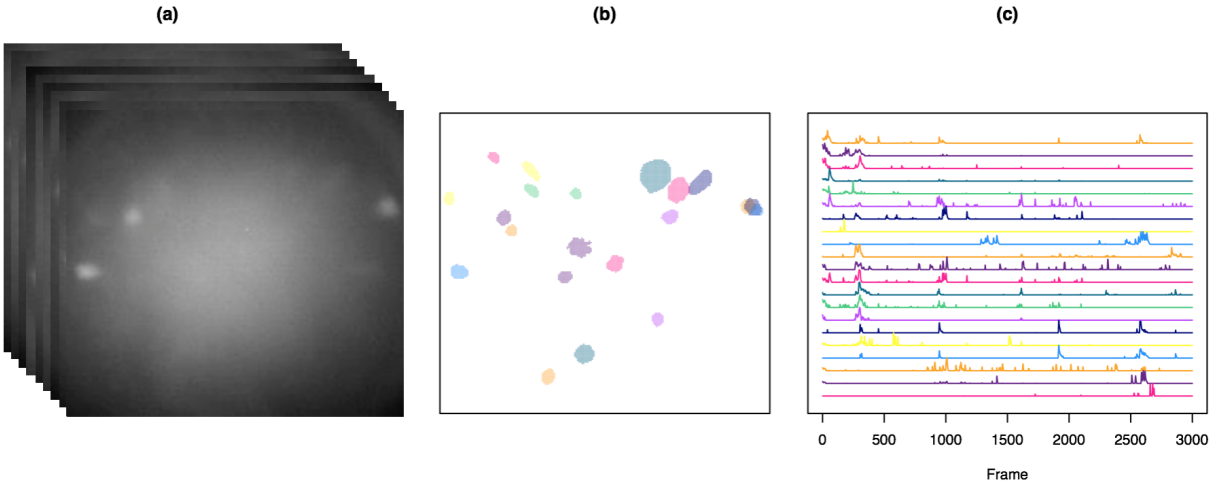


Figure 4.1: In (a), we display sample frames from the raw calcium imaging video. We wish to construct a spatial map of the neurons, like that shown in (b), and to estimate the calcium trace for each neuron over time, as shown in (c).

to more than one neuron. This step is sometimes referred to as *region of interest identification* or *cell sorting*. An example of neurons identified from a calcium imaging video is shown in Figure 4.1(b).

- *Calcium quantification:* The goal is to estimate the intracellular calcium concentration for each neuron during each frame of the movie. An example of these estimated *calcium traces* is shown in Figure 4.1(c).

To a certain extent, these two goals can be accomplished by visual inspection. However, visual inspection suffers from several shortcomings:

- It is subjective and it is not reproducible. Two people who view the same video may identify a different set of neurons or different firing times.

- It does not yield numerical information regarding neuron firing times, which may be needed for downstream analyses [Pillow et al., 2011].

- It may be inaccurate: for instance, a neuron that is very dim or that fires infrequently may not be identified by visual inspection.

- It is not feasible on videos with very large neuronal populations or very long durations. In fact, a typical calcium imaging video is 250,000 pixels and more than 50,000 frames, making visual inspection essentially impossible.

In this chapter, we propose a method that identifies the locations of neurons, and estimates their calcium concentrations over time. Previous proposals to automatically accomplish these tasks have been proposed in the literature, and are reviewed in Section 4.2. However, our method has several advantages over competing approaches. Unlike existing approaches, it:

- Involves few tuning parameters, which are themselves interpretable to the user, can for the most part be set to default values, and can be varied independently;

- Yields nested results when tuning parameters are varied;

- Is computationally feasible even on very large data sets; and

- Can be used to identify individual neurons, rather than sets of neurons, without post-processing.

The remainder of this chapter is organized as follows. We introduce notation in Section 4.1. In Section 4.2, we review related work. We present our proposal in Section 4.3, and discuss the selection of tuning parameters in Section 4.4. We apply our method to a calcium imaging video in Section 4.5. We discuss further details in Section 4.6. We close with a discussion in Section 4.7. Proofs are in Appendix C.

4.1 Notation

Let P denote the total number of pixels per image frame, and T the number of frames of the video. We define $\boldsymbol{Y}$ to be a $P \times T$ matrix for which the (i,j)th element, $y_{i,j}$, contains the fluorescence of the ith pixel in the jth frame. We let $\boldsymbol{y}_{i,\cdot} = \begin{pmatrix} y_{i,1} & y_{i,2} & \dots & y_{i,T} \end{pmatrix}^T$ represent the fluorescence of the ith pixel at each of the T frames. We let $\boldsymbol{y}_{\cdot,j} = \begin{pmatrix} y_{1,j} & y_{2,j} & \dots & y_{P,j} \end{pmatrix}^T$ represent the fluorescence of all P pixels during the jth frame. We use the same subscript conventions in order to reference the elements, rows, and columns of other matrices.

The goal of our work is to (1) identify the locations of the neurons; and (2) quantify calcium concentrations for these neurons over time. We view these tasks in the framework of a matrix factorization problem: we decompose $\boldsymbol{Y}$ into a matrix of spatial components, $\boldsymbol{A} \in \mathbb{R}^{P \times K}$, and a matrix of temporal components, $\boldsymbol{Z} \in \mathbb{R}^{K \times T}$, such that

$$\boldsymbol{Y} \approx \boldsymbol{A}\boldsymbol{Z}, \tag{4.1}$$

where K is the total number of estimated neurons. Note that $\boldsymbol{a}_{\cdot,k}$ specifies which of the P pixels of the image frame are mapped to the kth neuron, and $\boldsymbol{z}_{k,\cdot}$ quantifies the calcium concentration for the kth neuron at each of the T video frames. The number of neurons K is unknown, and must be determined as part of the analysis.

4.2 Related Work

We focus on methods proposed to accomplish both goals: identifying neurons and quantifying calcium concentrations.

One of the first automatic methods in this area was proposed by Mukamel et al. [2009]. This method first applies principal component analysis to reduce the dimensionality of the data, followed by spatio-temporal independent component analysis to produce spatial and temporal components that are statistically independent of one another. Though this method is widely used, it often requires heuristic post-processing of the spatial components, and typically fails to distinguish between spatially overlapping neurons [Pnevmatikakis et al., 2016].

To better handle overlapping neurons, Maruyama et al. [2014] proposed a non-negative matrix factorization approach, which estimates $\boldsymbol{A}$ and $\boldsymbol{Z}$ in (4.1) by solving

$$\underset{\boldsymbol{A}\geq\boldsymbol{0},\boldsymbol{Z}\geq\boldsymbol{0},\boldsymbol{a}_b\geq\boldsymbol{0}}{\text{minimize}} \quad \frac{1}{2}\|\boldsymbol{Y}-\boldsymbol{A}\boldsymbol{Z}-\boldsymbol{a}_b\boldsymbol{z}_b^T\|_F^2. \tag{4.2}$$

In (4.2), the term $\boldsymbol{a}_b\boldsymbol{z}_b^T$ is a rank-one correction for background noise: $\boldsymbol{z}_b \in \mathbb{R}^T$ is a temporal representation of the background noise (known as the *bleaching line*, and estimated using a linear fit to average fluorescence over time of a background region), and $\boldsymbol{a}_b \in \mathbb{R}^P$ is a spatial representation of the background noise. Element-wise positivity constraints are imposed on $\boldsymbol{A}$, $\boldsymbol{Z}$, and $\boldsymbol{a}_b$ in (4.1). While (4.2) can handle overlapping neurons, there are no constraints on the sparsity of $\boldsymbol{A}$ or $\boldsymbol{Z}$, or locality constraints on $\boldsymbol{A}$. Thus, the estimated temporal components are very noisy, and the estimated spatial components are often not localized and must be post-processed heuristically.

To overcome these shortcomings, Haeffele et al. [2014] modify (4.2) so that the temporal components $\boldsymbol{A}$ are sparse and the spatial components $\boldsymbol{Z}$ are sparse and have low total variation. Recently, Pnevmatikakis et al. [2016] further refine (4.2) by explicitly modeling the dynamics of the calcium when estimating the temporal components $\boldsymbol{Z}$, and combining a sparsity constraint with intermediate image filtering when estimating the spatial components $\boldsymbol{A}$.

Related approaches are taken by Apthorpe et al. [2016], Diego et al. [2013], Diego Andilla and Hamprecht [2013, 2014], Friedrich et al. [2015], Pnevmatikakis and Paninski [2013], Zhou et al. [2016].

While these existing approaches show substantial promise, and are a marked improvement over visual identification of neurons from the calcium imaging videos, they also suffer from some shortcomings:

- The optimization problems (see, e.g., (4.2)) are biconvex. Thus, algorithms typically get trapped in local optima. Furthermore, the results strongly depend on the choice of initialization.

- Each method involves several user-selected tuning parameters. There is no natural interpretation to these tuning parameters, which leads to challenges in selection. Furthermore, changing one tuning parameter may necessitate updating all of them. Moreover, there is no natural nesting with respect to the tuning parameters: a slight increase or decrease in one tuning parameter can lead to a completely different set of identified neurons.

- The number of neurons K must be specified in advance, and the estimates obtained for different values of K will not be nested: two different values of K can yield completely different answers.

- Post-processing of the identified neurons is often necessary.

- Implementation on very large data sets can be computationally burdensome.

We illustrate the difficulties of existing matrix factorization approaches with a toy example.

To overcome these challenges, instead of simultaneously estimating $\boldsymbol{A}$ and $\boldsymbol{Z}$ in the model (4.1), we propose a two-stage procedure. In the first stage, we leverage spatial information in order to obtain a very good estimate of the spatial components $\boldsymbol{A}$. In the second stage, we use our estimate of $\boldsymbol{A}$ in order to obtain an accurate estimate of the temporal components $\boldsymbol{Z}$. This two-stage approach allows us to re-cast (4.1), a very challenging unsupervised learning problem, into a much easier supervised learning problem. Compared to existing approaches, our proposal is much faster to solve computationally, involves more interpretable tuning parameters, and yields substantially more accurate results.

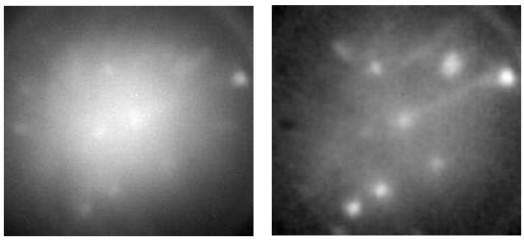
4.3 Proposed Approach

In Figure 4.2, we summarize the proposed method, Segmentation, Clustering, and Lasso Penalties (SCALPEL), which consists of four stages:

Step 0. Data Pre-Processing (Appendix A.1)

- Perform spatial and temporal smoothing
- Remove bleaching effect
- Standardize using $\Delta f/f$ transformation

Figures (a) and (b) illustrate the results of Step 0 for one of the 3000 frames in the calcium imaging video.

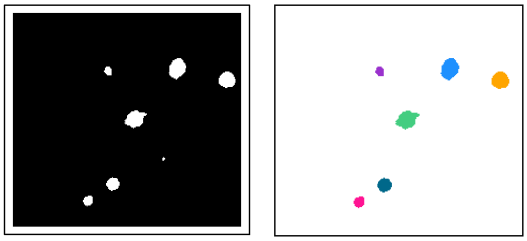


(a) Original frame (b) Pre-processed frame

Step 1. Construction of a Spatial Component Dictionary (Section 4.1)

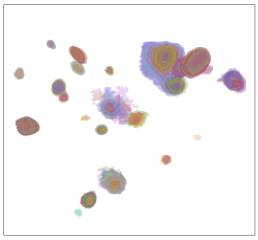
- Threshold image
- Identify connected components
- Filter components

Figures (c) and (d) illustrate the results of Step 1 for the same frame shown in Figures (a) and (b).



(c) Thresholded frame (d) Frame's candidate spatial components

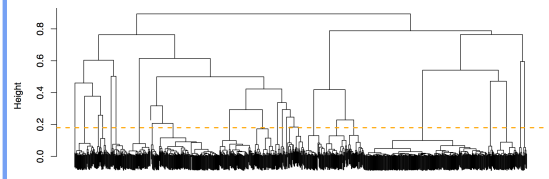
Figure (e) shows the resulting candidate spatial components for all 3000 frames.



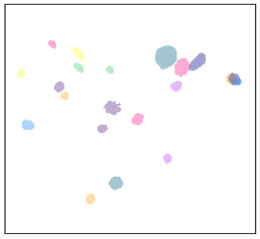
(e) Movie's 839 candidate spatial components

Step 2. Refinement of the Spatial Component Dictionary (Section 4.2)

- Calculate dissimilarity matrix
- Perform prototype clustering
- Discard small clusters



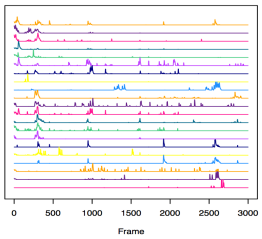
(f) Dendrogram resulting from prototype clustering with cut-point shown in orange



(g) The 21 representative spatial components

Step 3. Spatial Component Selection and Temporal Component Estimation (Sections 4.3 and 8)

- Solve sparse group lasso problem with non-negativity constraint



(h) Temporal components corresponding to spatial components in (g)

Figure 4.2: A summary of the SCALPEL procedure.

Step 0. *Data Pre-Processing:* We apply standard pre-processing techniques in order to smooth the data both temporally and spatially, remove the bleaching effect, and calculate a standardized fluorescence. Details are provided in Appendix C.1. In what follows, $\boldsymbol{Y}$ refers to the calcium imaging data after these three pre-processing steps have been performed.

Step 1. *Construction of a Spatial Component Dictionary:* We apply a simple image segmentation procedure to each frame of the video in order to derive a dictionary of candidate spatial components. Ideally, this dictionary is a superset of the true spatial components. This is discussed further in Section 4.3.1.

Step 2. *Refinement of the Spatial Component Dictionary:* We eliminate redundancy in the spatial component dictionary by clustering together candidate spatial components that represent the same neuron, based on spatial and temporal information. This results in a matrix $\boldsymbol{A} \in \mathbb{R}^{P \times K}$, with binary elements; $a_{j,k} = 1$ if the jth pixel is contained in the kth refined candidate spatial component. More details are discussed in Section 4.3.2.

Step 3. *Spatial Component Selection and Temporal Component Estimation:* We estimate the temporal components $\boldsymbol{Z}$ corresponding to the refined set of spatial components by solving a sparse group lasso problem with a non-negativity constraint. The kth row of $\hat{\boldsymbol{Z}}$ is the estimated calcium trace corresponding to the kth spatial component. For some values of the tuning parameters, rows of $\hat{\boldsymbol{Z}}$ may be entirely equal to zero. Thus spatial components are selected while the temporal components are estimated. Additional details are presented in Sections 4.3.3 and 4.6.

Throughout this section, we illustrate SCALPEL on a calcium imaging data set that has 205×226 pixels and 3000 frames. Figures 4.1-4.9, as well as Figures C.1, C.2, and C.4 in Appendix C, involve this data set. Full analysis of this data set is detailed in Section 4.5.

4.3.1 Step 1: Construction of a Spatial Component Dictionary

In this step, we identify a large set of candidate neurons. We do this by performing a simple image segmentation procedure on each frame separately. Consequently, this step can be easily parallelized across the frames of lengthy videos. The procedure is as follows:

1. *Threshold Image:* We create a binary image by thresholding the image frame using the 99.9% quantile of $\boldsymbol{Y}$, where the quantile is computed using the entire video. In Section 4.4.1, we discuss alternative quantile choices for thresholding. Figure 4.3(b) displays the binary image that results from thresholding the frame shown in Figure 4.3(a).

2. *Identify Connected Components:* We identify the connected components of the white pixels of the thresholded image. Some of these connected components may represent neurons, whereas others are likely to be artifacts or else multiple neurons located near each other that fire simultaneously in a frame. We use 4-connectivity to define connected components, though 8-connectivity would also be reasonable.

3. *Filter Components:* To eliminate noise, we perform two steps:

   (a) We discard connected components consisting of fewer than 25 pixels. Due to their very small size, these are unlikely to be true neurons.

   (b) For each pixel in a connected component, we calculate the number of its neighbors (out of 8 total neighbors) that are contained in that connected component. We then calculate the average number of neighbors per connected component. We discard connected components containing fewer than 100 pixels and fewer than 5.5 average neighbors. We also discard connected components containing more than 100 pixels and fewer than 6 average neighbors. This criterion is size-based, as smaller components have a greater proportion of edge pixels and thus a lower average number of neighbors.

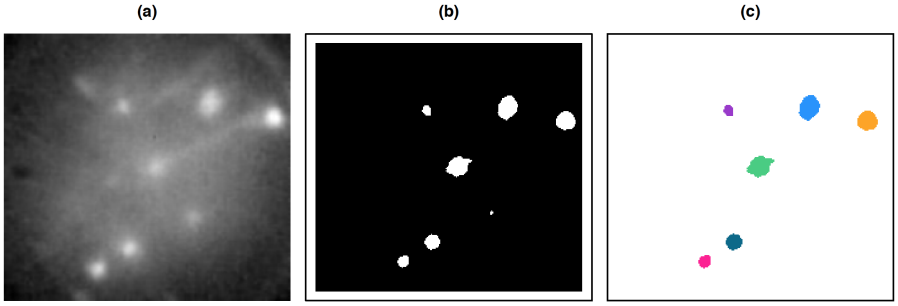


Figure 4.3: In (a), we display a single frame of a calcium imaging video after performing the pre-processing described in Appendix C.1. In (b), we show the binary image that results after thresholding using the 99.9% quantile of the video's elements. In (c), we display the six connected components from the image in (b) that contain at least 25 pixels.

The K^0 connected components that remain after this filtering step are the candidate spatial components for our dictionary, which we use to construct the matrix $\boldsymbol{A}^0 \in \mathbb{R}^{P \times K^0}$. That is, the kth column of $\boldsymbol{A}^0$ is a vector of 1's and 0's, indicating whether each pixel is contained in the kth candidate spatial component.

4.3.2 Step 2: Refinement of the Spatial Component Dictionary

The spatial component dictionary obtained in Step 1 contains a very large set of candidate spatial components: a single neuron will appear in the dictionary many times, once per frame in which it is active; furthermore, some candidate spatial components will be the result of noise artifacts. To refine the dictionary, we wish to (1) cluster candidate spatial components that correspond to the same neuron, and identify a representative spatial component for each cluster; and (2) discard noise candidate spatial components, which will cluster with very few other candidate spatial components. We proceed as follows:

1. *Calculate Dissimilarity Matrix:* We use a novel dissimilarity metric, which incorporates both spatial and temporal information, in order to calculate the dissimilarity between every pair of candidate spatial components. More details are given below.

2. *Perform Prototype Clustering:* We use the aforementioned pair-wise dissimilarities in order to perform *prototype clustering* of candidate spatial components [Bien and Tibshirani, 2011]. This clustering also yields a representative spatial component for each cluster. More details are given below.

3. *Discard Small Clusters:* We discard representative spatial components corresponding to clusters containing fewer than five candidate spatial components, as these clusters likely are due to noise artifacts. The remaining representative spatial components make up the columns of the spatial component matrix $\boldsymbol{A}$, which will be used in Step 3, discussed in Section 4.3.3.

Choice of Dissimilarity Metric

Before performing clustering, we must decide how to quantify similarity between the K^0 candidate spatial components obtained in Step 1. Candidate spatial components that correspond to the same neuron are likely to have (1) similar spatial maps and (2) similar average fluorescence over time. To this end, we construct a dissimilarity metric that leverages both spatial and temporal information.

We define $p_{i,j} = (\boldsymbol{a}^0_{\cdot,i})^T \boldsymbol{a}^0_{\cdot,j}$, the number of pixels shared between the ith and jth candidate spatial components. When $i = j$, $p_{i,i}$ is simply the number of pixels in the component. We then define the spatial dissimilarity between the ith and jth candidate spatial components to be

$$d^s_{i,j} = 1 - \frac{p_{i,j}}{\sqrt{p_{i,i} p_{j,j}}}. \tag{4.3}$$

Thus, $d^s_{i,j} = 1$ if and only if the ith and jth spatial components are non-overlapping, and $d^s_{i,j} = 0$ if and only if the ith and jth spatial components are identical. Note that $d^s_{i,j}$ is known as the cosine dissimilarity or Ochiai coefficient [Gower, 2006]. Alternatives to (4.3) are discussed in Appendix C.2.

We define the temporal dissimilarity between the ith and jth candidate spatial compo-

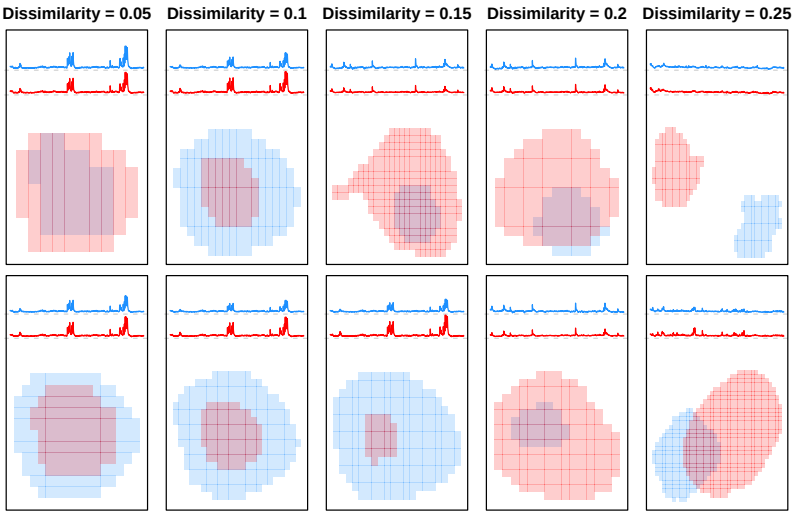


Figure 4.4: Each column displays two pairs of candidate spatial components having overall dissimilarities, as defined in (4.4), of 0.05, 0.1, 0.15, 0.2, and 0.25. For each candidate spatial component, the average standardized fluorescence over time and the (zoomed-in) spatial map are shown.

nents as

$$d^t_{i,j} = 1 - (\text{corr}(\boldsymbol{Y}^T \boldsymbol{a}^0_{\cdot,i}, \boldsymbol{Y}^T \boldsymbol{a}^0_{\cdot,j}))_+,$$

where $d_+ \equiv \max(0, d)$ denotes the positive part operator. Note that $\boldsymbol{Y}^T \boldsymbol{a}^0_{\cdot,i} \in \mathbb{R}^T$ is the fluorescence summed over all pixels in the ith candidate spatial component, $\boldsymbol{a}^0_{\cdot,i}$, at each time frame.

Finally, the overall dissimilarity is

$$d_{i,j} = \omega d^s_{i,j} + (1 - \omega) d^t_{i,j}, \tag{4.4}$$

where $\omega \in [0, 1]$ controls the relative weightings of the spatial and temporal information. In Figure 4.4, we illustrate pairs of candidate spatial components with various dissimilarities for $\omega = 0.2$. We use $\omega = 0.2$ to obtain the results shown throughout this chapter.

Prototype Clustering

We now consider the task of clustering the candidate spatial components. Because we do not wish to pre-select the number of clusters, and because we want the solutions to be nested as the number of clusters is varied, we opt to use hierarchical clustering.

In order to perform hierarchical clustering, we must specify both a dissimilarity measure and a type of linkage [Hastie et al., 2009]. We will use the dissimilarity given in (4.4). Because standard linkages, such as complete, average, or single linkage, will not provide a representative spatial component associated with each cluster, we opt to use *minimax* linkage, introduced in Bien and Tibshirani [2011]. The minimax linkage distance between two clusters C_1 and C_2 is given by

$$\min_{\boldsymbol{a}^0_{\cdot,i} \in C} \max_{\boldsymbol{a}^0_{\cdot,j} \in C} d_{i,j}, \tag{4.5}$$

where $C \equiv C_1 \cup C_2$. In (4.5), each cluster is represented by a single candidate spatial component, or prototype, which has the smallest maximum dissimilarity to all of the other candidate spatial components in the cluster. Consequently, Bien and Tibshirani [2011] refer to the use of (4.5) as *prototype clustering*. We will use each cluster's prototype, which we refer to as the representative spatial component, to represent its spatial components. These K representative spatial components are the columns of the spatial component matrix, $\boldsymbol{A} \in \mathbb{R}^{P \times K}$, with binary elements; $a_{j,k} = 1$ if the jth pixel is contained in the kth representative spatial component.

We apply prototype clustering to the calcium imaging data set that we have been considering using the `R` package `protoclust` [Bien and Tibshirani, 2015]. The resulting dendrogram is shown in Figure 4.5(a). In Section 4.4.2, we discuss how to choose a cut-point, or height at which to cut the dendrogram. Results for different cut-points are displayed in Figures 4.5(b)–(e). An example of how the representative spatial component relates to the other candidate spatial components in the cluster is given in Appendix C.3.

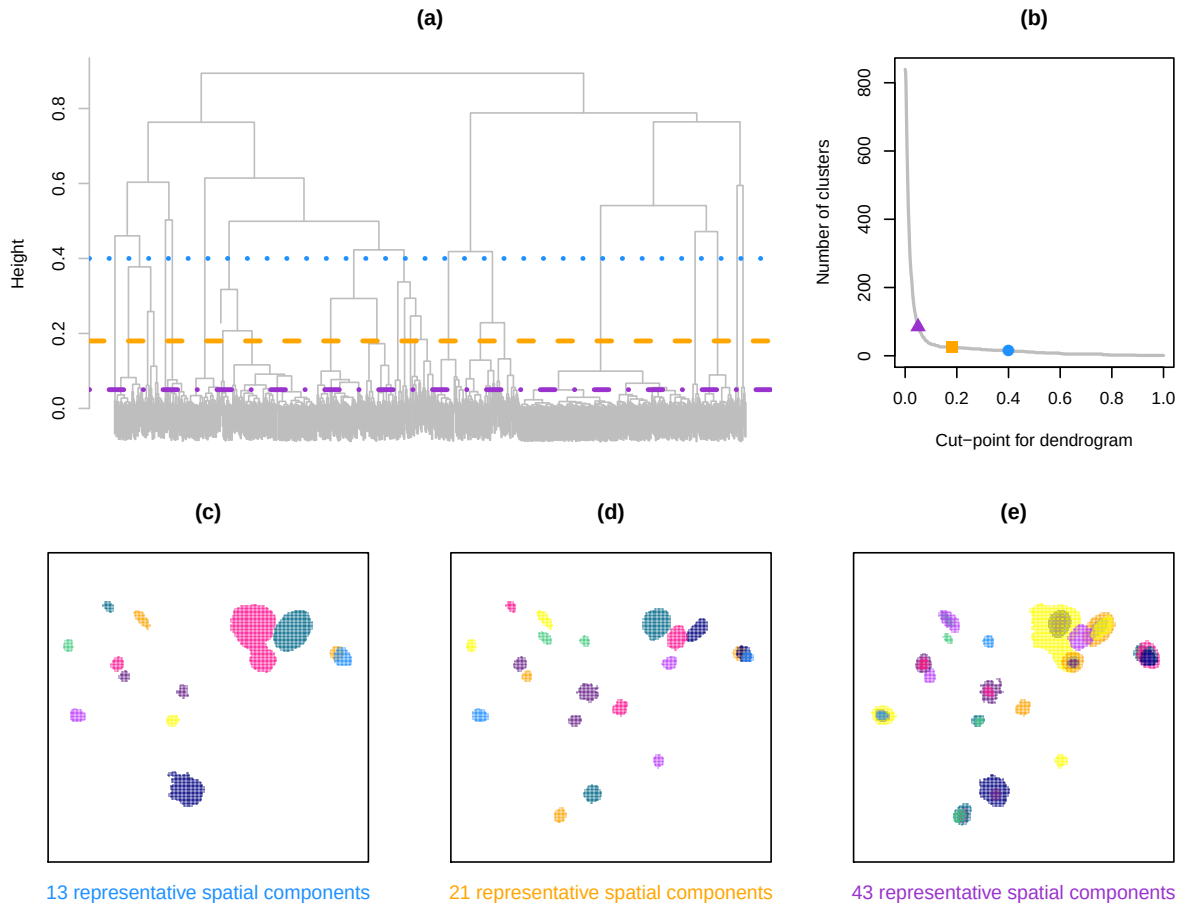


Figure 4.5: In (a), we plot the dendrogram that results from prototype clustering, and indicate three different cut-points on the tree, 0.05 (–·–), 0.18 (– –), and 0.4 (····). In (b), we plot the number of clusters that result from cutting the dendrogram in (a) at different heights. In (c)–(e), we show the representative spatial components that result from using cut-points of 0.4, 0.18, and 0.05, respectively.

4.3.3 Step 3: Spatial Component Selection and Temporal Component Estimation

We are now left with K representative spatial components, many of which are likely to be neurons. In this final step, we estimate the temporal components associated with these representative spatial components. We do this by solving a sparse group lasso problem with a non-negativity constraint,

$$\underset{\boldsymbol{Z}\in\mathbb{R}^{K\times T},\boldsymbol{Z}\geq 0}{\text{minimize}} \quad \frac{1}{2}\left\|\boldsymbol{Y}-\tilde{\boldsymbol{A}}\boldsymbol{Z}\right\|_F^2+\lambda\alpha\mathbf{1}^T\boldsymbol{Z}\mathbf{1}+\lambda(1-\alpha)\sum_{k=1}^{K}\left\|\boldsymbol{z}_{k,\cdot}\right\|_2, \tag{4.6}$$

where $\alpha\in[0,1]$ and $\lambda>0$ are tuning parameters, and $\tilde{\boldsymbol{A}}$ is defined as $\tilde{\boldsymbol{a}}_{\cdot,k}=\boldsymbol{a}_{\cdot,k}/\|\boldsymbol{a}_{\cdot,k}\|_2^2$. The justification for this scaling of $\boldsymbol{A}$ is given in Section 4.6.3.

The first term of the objective in (4.6) encourages the spatiotemporal factorization (4.1) to fit the data closely. The second term in (4.6) encourages the temporal components to be sparse, which is desirable as each neurons is only active in a small number of frames. Note that due to the element-wise non-negativity constraint on $\boldsymbol{Z}$, $\mathbf{1}^T\boldsymbol{Z}\mathbf{1}=\sum_{k=1}^{K}\left\|\boldsymbol{z}_{k,\cdot}\right\|_1$, which is the sparsity-inducing lasso penalty [Tibshirani, 1996]. The third term in (4.6) encourages group-wise sparsity on the rows of $\boldsymbol{Z}$ [Yuan and Lin, 2006], which leads to selection of the representative spatial components: for $\lambda(1-\alpha)$ sufficiently large, only a subset of the K representative spatial components will have a non-zero temporal component.

In Section 4.6, we discuss our algorithm for solving (4.6), justify the scaling of $\boldsymbol{A}$ used in (4.6), and present conditions under which the solution to (4.6) is sparse. Readers simply interested in the practical use of our method may disregard this section.

4.4 Tuning Parameter Selection

Many competing methods for identifying neurons in calcium imaging data involve several tuning parameters, which when varied can have a huge impact on the results. When one tuning parameter is changed, the others may be also need to be updated. Adding to the challenge, the solutions corresponding to different tuning parameters are not necessarily nested, meaning that small changes in tuning parameters could lead to substantially different

results.

SCALPEL involves a number of tuning parameters:

- *Step 1:* Quantile threshold for image segmentation
- *Step 2:* Cut-point for dendrogram
- *Step 3:* λ and α for Equation 4.6

However, in marked contrast to competing methods, the tuning parameters in SCALPEL can be chosen independently at each step. Furthermore, we strongly recommend the use of default values for all but the choice of λ in Step 3. Therefore, in practice, SCALPEL only involves one tuning parameter that must be selected by the user. We discuss this at greater length below.

4.4.1 Tuning Parameter for Step 1

The tuning parameter in Step 1 is the quantile threshold used in image segmentation. We have used a 99.9% quantile threshold on all of the calcium imaging data sets that we have considered. In principle, this quantile threshold may need to be adjusted; however, it is straightforward to select a reasonable threshold by visually examining a few frames and their corresponding binary images, as in Figures 4.3(a) and (b).

4.4.2 Tuning Parameter for Step 2

In Step 2, we must choose a height $h \in [0, 1]$ at which to cut the dendrogram that results from hierarchical clustering of the candidate spatial components, as shown in Figure 4.5(a). Fortunately, the cut-point has a nice interpretation, which can help guide our choice. Cutting the dendrogram at a height of h guarantees that each cluster's prototype has a dissimilarity no more than h with each of the candidate spatial components in that cluster. Figure 4.4 displays pairs of candidate spatial components with a given dissimilarity. On the data sets

that we have analyzed, cut-points between 0.15 and 0.2 are appropriate. The investigator can either choose a cut-point in this range by visual inspection of the resulting representative spatial components, or can simply fix the cut-point (for example, at 0.15) regardless of the data set.

4.4.3 Tuning Parameters for Step 3

In Step 3, we must choose values of λ and α in (4.6). We have had empirical success using a fixed value of α near 1, as we expect each neuron to spike a small number of times. We used $\alpha = 0.9$ to obtain all of the results in this chapter.

To select λ, we can use cross-validation or a training/validation set approach, as illustrated in Section 4.5 and detailed in Appendix C.4. In fact, because (4.6) is a supervised learning problem, cross-validation is a natural option. Performing cross-validation is computationally feasible, because solving (4.6) is actually quite efficient, as detailed in Section 4.6.2. Furthermore, Corollary 4.6.5 makes it easy to determine the largest value of λ that should be considered in our cross-validation scheme.

Alternatively, we can choose the value of λ by visual inspection of the calcium traces, like those shown in Figure 4.1(c), at each value of λ.

4.5 Results for Calcium Imaging Data

We now present the results from applying SCALPEL to the calcium imaging video used as an example in Section 4.3. This video has 3000 frames of size 205×226 pixels. Using a quantile threshold of 99.9% quantiile, Step 1 of SCALPEL resulted in 839 candidate spatial components, which were then clustered into 25 clusters corresponding to a cut-point of 0.18 in Step 2. Four of these clusters had fewer than 5 candidate spatial components assigned to them, and thus were discarded. In Step 3, we used a training and validation sets approach, as described in Appendix C.4, to choose the value of λ. The results are shown in Figure 4.6. In Figure 4.7, we compare the estimated neurons to a pixel-wise variance plot of the calcium imaging video. We expect pixels that are part of true neurons to have higher variance than

pixels not associated with any neurons. Indeed, we see that many of the estimated neurons coincide with regions of high pixel-wise variance. However, some estimated neurons are in regions with low variance. In Figure 4.8(a), we provide evidence that these estimated neurons may still be true neurons. In Figures 4.8(b) and (c), we illustrate the impact that undetected neurons have on the estimated calcium concentrations of neighboring detected neurons. In Figure 4.9, we illustrate that it may be desirable to choose λ to be larger than that chosen via a training and validation sets approach. Choosing a larger value of λ will result in greater sparsity of the temporal component estimates, which is consistent with the scientific context. Choosing λ via a training and validation sets approach will likely result in a smaller λ than if λ is chosen empirically, because the goal scientifically is to model the brightest areas of the video well, whereas a training and validation sets approach aims to find a value of λ that models the entire video well, both bright areas and the background. The less residual background effects that remain after pre-processing, the closer that the λ chosen via training and validation sets will be to the value chosen empirically.

4.6 Further Discussion of Step 3

In this section, we elaborate on issues related to solving the sparse group lasso problem (4.6) in Step 3. The discussion in this section involves technical details, and can be skipped by readers only interested in the practical use of SCALPEL. We discuss the solution to (4.6) when $K = 1$ in Section 4.6.1, our algorithm for solving (4.6) for any value of K in Section 4.6.2, the justification for the scaling of $\boldsymbol{A}$ in Section 4.6.3, and a result about the tuning parameters α and λ that lead to a sparse solution in Section 4.6.4.

4.6.1 Single Component Problem

We first consider solving (4.6) in the setting in which there is a single spatial component ($K = 1$). While calcium imaging data will not have only a single neuron, this setting provides intuition, and will prove useful when we later solve (4.6) for $K > 1$ in Section 4.6.2.

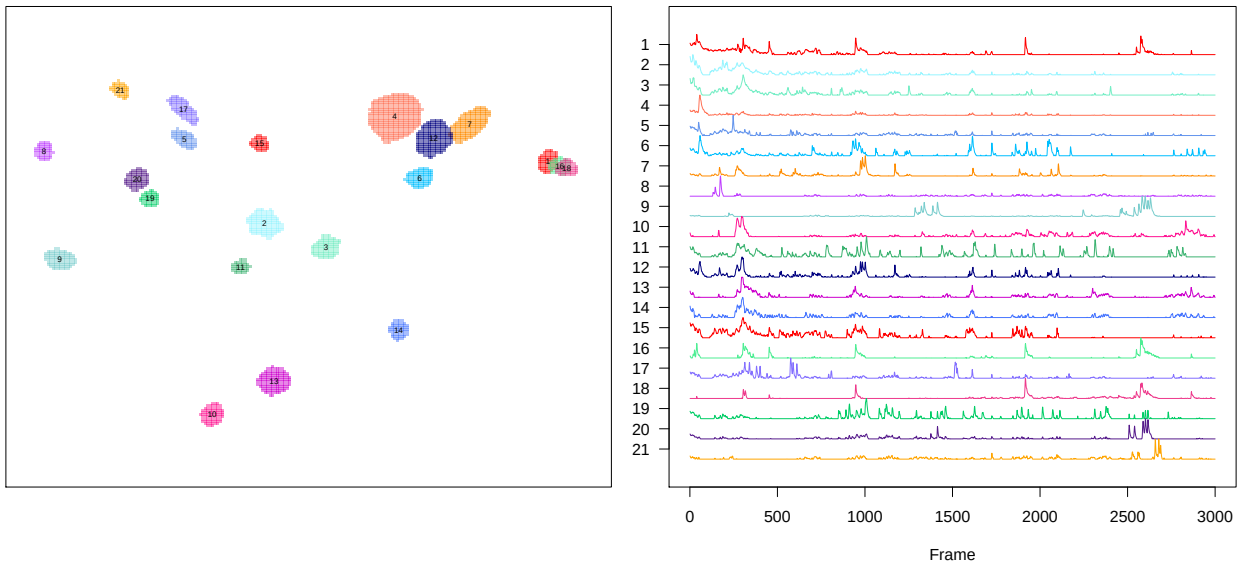


Figure 4.6: We plot the spatial maps for the 21 representative spatial components, along with their estimated intracellular calcium concentrations corresponding to λ chosen via a training and validation sets approach.

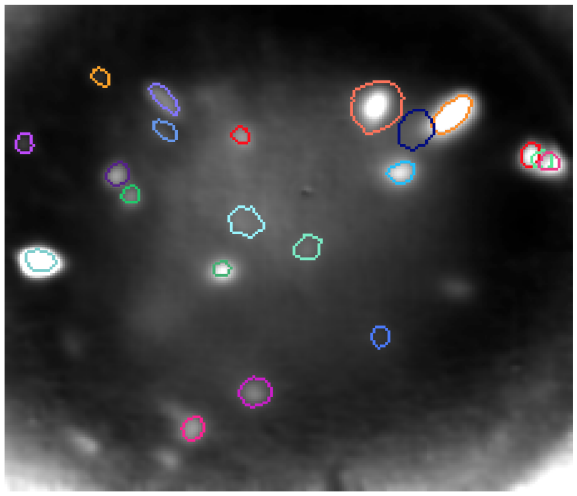


Figure 4.7: We compare the outlines of the estimated neurons (also shown in Figure 4.6) to a heat map of the pixel-wise variance of the calcium imaging video. That is, we plot the variance of each pixel over the 3000 frames, with whiter points indicating higher variance.

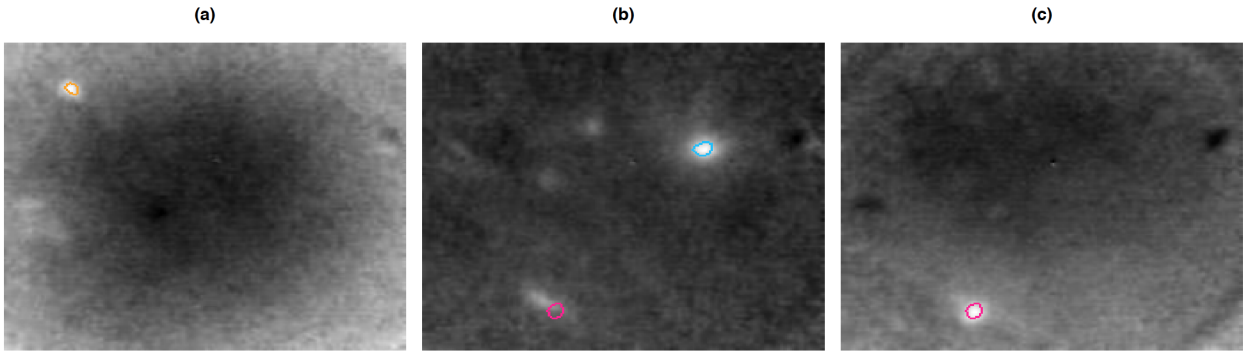


Figure 4.8: In (a), we see that one of the estimated neurons does correspond to a true neuron, even though it was shown in Figure 4.7 to be in a low-variance region. In (b), we see a frame in which one of the estimated neurons is erroneously estimated to have a non-zero calcium concentration due to a neighboring, undetected neuron being active. In (c), we see a frame in which that same estimated neuron is correctly estimated to have a non-zero calcium concentration.

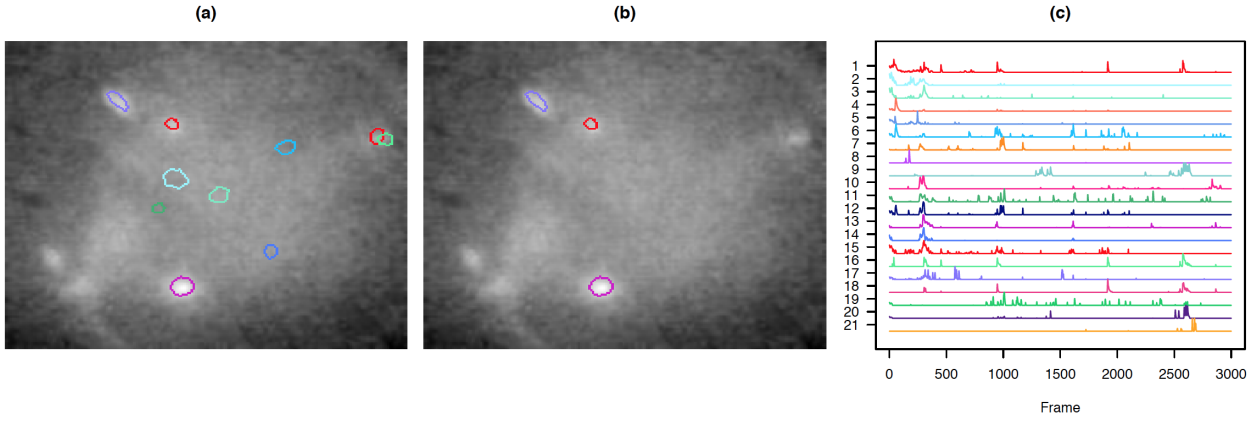


Figure 4.9: In (a), for an example frame, we show the estimated neurons with non-zero calcium concentrations for λ chosen via a training and validation sets approach. In (b), for the same frame, we show the estimated neurons with non-zero calcium concentrations for λ chosen empirically. The full temporal component results for λ chosen empirically are shown in (c).

Lemma 4.6.1 *The solution to*

$$\underset{\boldsymbol{z}\in\mathbb{R}^T\geq\mathbf{0}}{\text{minimize}} \quad \frac{1}{2}\left\|\boldsymbol{Y}-\tilde{\boldsymbol{a}}\boldsymbol{z}^T\right\|_F^2+\lambda\alpha\mathbf{1}^T\boldsymbol{z}+\lambda(1-\alpha)\left\|\boldsymbol{z}\right\|_2$$

is

$$\hat{\boldsymbol{z}}=\left(1-\frac{\lambda(1-\alpha)}{\left\|(\boldsymbol{Y}^T\tilde{\boldsymbol{a}}-\lambda\alpha\mathbf{1})_+\right\|_2}\right)_+\left(\frac{\boldsymbol{Y}^T\tilde{\boldsymbol{a}}-\lambda\alpha\mathbf{1}}{\tilde{\boldsymbol{a}}^T\tilde{\boldsymbol{a}}}\right)_+, \tag{4.7}$$

where the positive part operator is applied element-wise.

The proof of Lemma 4.6.1 is in Appendix C.5. We can inspect the solution (4.7) to gain intuition. Recall that $\tilde{\boldsymbol{a}}_{\cdot,k}\equiv\boldsymbol{a}_{\cdot,k}/\|\boldsymbol{a}_{\cdot,k}\|_2^2$, where $\boldsymbol{a}_{\cdot,k}$ has binary elements. Therefore, $\boldsymbol{Y}^T\tilde{\boldsymbol{a}}\in\mathbb{R}^T$ is the average fluorescence of pixels in the spatial component at each of the frames and $\frac{1}{\tilde{\boldsymbol{a}}^T\tilde{\boldsymbol{a}}}$ equals the number of pixels in the spatial component. When $\lambda=0$, $\hat{\boldsymbol{z}}=\left(\frac{\boldsymbol{Y}^T\tilde{\boldsymbol{a}}}{\tilde{\boldsymbol{a}}^T\tilde{\boldsymbol{a}}}\right)_+$, which is simply the positive part of the total fluorescence at all pixels in the spatial component over time. We now consider the impact of λ for three different settings of α:

- $\alpha=1$*:* In this setting, $\hat{\boldsymbol{z}}$ is the positive part of the soft-thresholded total fluorescence. This encourages elements of $\hat{\boldsymbol{z}}$ to exactly zero for frames in which the spatial component has low fluorescence.

- $\alpha=0$*:* In this setting, $\hat{\boldsymbol{z}}$ is found by scaling all elements of the total fluorescence by the same amount. Thus, individual elements of $\hat{\boldsymbol{z}}$ are not encouraged to be 0, though $\hat{\boldsymbol{z}}=\mathbf{0}$ if the spatial component has a low amount of fluorescence across all frames (i.e., $\|\boldsymbol{Y}^T\tilde{\boldsymbol{a}}\|_2$ small) or λ is very large.

- $\alpha\in(0,1)$*:* Both soft-thresholding and soft-scaling are performed, which encourages sparsity of individual elements of $\hat{\boldsymbol{z}}$ and the entire vector $\hat{\boldsymbol{z}}$, respectively.

In Figure 4.10, we illustrate the values of $\hat{\boldsymbol{z}}$ for the three scenarios described above.

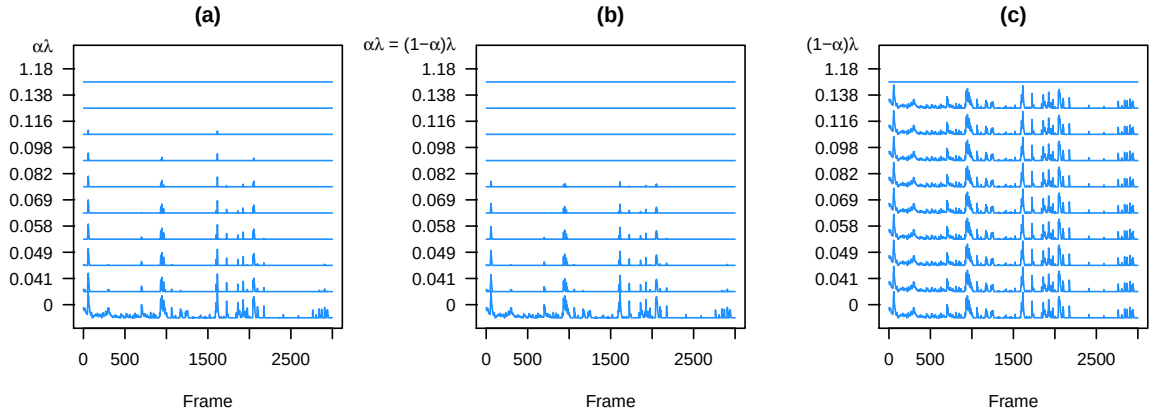


Figure 4.10: We plot the solution $\hat{\boldsymbol{z}}$, as given in (4.7), for a range of λ when (a) only a lasso penalty is used ($\alpha = 1$), (b) a mixture of penalties is used ($\alpha = 0.5$), and (c) only a group lasso penalty is used ($\alpha = 0$).

4.6.2 Algorithm

We now consider how to solve (4.6) for $K > 1$. While generalized gradient descent [Beck and Teboulle, 2009] can be used to solve concurrently for $\boldsymbol{z}_{1,\cdot}, \ldots, \boldsymbol{z}_{K,\cdot}$ in (4.6), the problem is solved more efficiently by noting that (4.6) is decomposable into groups of overlapping spatial components.

Let $\mathcal{N}_1, \ldots, \mathcal{N}_S$ denote a partition of the K representative spatial components, such that $\mathcal{N}_s \cap \mathcal{N}_{s'} = \emptyset$ for $s \neq s'$, and $\cup_{s=1}^{S} \mathcal{N}_s = \{1, \ldots, K\}$. Define the mapping $\mathcal{M}(\mathcal{N}_s) = \{p \in (1, \ldots, P) : (\tilde{\boldsymbol{A}}_{p,\mathcal{N}_s})^T \mathbf{1} > 0\}$. That is, $\mathcal{M}(\mathcal{N}_s)$ indexes the set of pixels that are active in that subset of neurons.

Lemma 4.6.2 *Suppose that $\mathcal{M}(\mathcal{N}_s) \cap \mathcal{M}(\mathcal{N}_{s'}) = \emptyset$ for all $s \neq s'$, so that there is no spatial overlap between the sets of representative spatial components $\mathcal{N}_1, \ldots, \mathcal{N}_S$. Then solving* (4.6) *gives the same solution as solving*

$$\underset{\boldsymbol{Z}_{\mathcal{N}_s,\cdot} \geq 0}{\text{minimize}} \quad \frac{1}{2} \left\| \boldsymbol{Y}_{\mathcal{M}(\mathcal{N}_s),\cdot} - \tilde{\boldsymbol{A}}_{\mathcal{M}(\mathcal{N}_s),\mathcal{N}_s} \boldsymbol{Z}_{\mathcal{N}_s,\cdot} \right\|_F^2 + \lambda\alpha \mathbf{1}^T \boldsymbol{Z}_{\mathcal{N}_s,\cdot} \mathbf{1} + \lambda(1-\alpha) \sum_{n \in \mathcal{N}_s} \|\boldsymbol{z}_{n,\cdot}\|_2 , \quad (4.8)$$

for $s = 1, 2, \ldots, S$.

The proof of Lemma 4.6.2 is in Appendix C.6.

Our approach to solving (4.8) depends on the size of $\mathcal{N}_s$. For $|\mathcal{N}_s| = 1$, we can simply use the closed-form solution for $\boldsymbol{z}_{\mathcal{N}_s,\cdot}$ given by Lemma 4.6.1. This is advantageous as the calcium imaging data sets that we have analyzed often have representative spatial components that do not overlap any others. For $|\mathcal{N}_s| > 1$, we use generalized gradient descent to solve for the global optimum of (4.8) [Beck and Teboulle, 2009].

In light of Lemma 4.6.2, in order to solve (4.6), we first partition the representative spatial components into S sets, $\mathcal{N}_1, \ldots, \mathcal{N}_S$, such that there is no overlap between the pixels in the S sets, and so that no set can be partitioned further. This can be done quickly, as outlined in Step 1 of Algorithm 4. Then, we solve (4.8) for $s = 1, \ldots, S$. Details are provided in Algorithm 4.

We typically solve (4.6) for a sequence of exponentially decreasing λ values. To improve computational performance, Step 2(b) of Algorithm 4 can be implemented using warm starts, in which $\boldsymbol{Z}^{(0)}_{\mathcal{N}_s,\cdot}$ is initialized as the solution for $\boldsymbol{Z}_{\mathcal{N}_s,\cdot}$ for the previous value of λ. Additional details regarding the derivation of the generalized gradient descent algorithm used in Step 2(b) of Algorithm 4 are given in Appendix C.7.

4.6.3 Scaling of $\boldsymbol{A}$

In (4.6), the kth column of the matrix $\tilde{\boldsymbol{A}}$ encodes the spatial mapping of the kth representative spatial component, after scaling. To obtain $\tilde{\boldsymbol{A}}$, we divide the kth column of $\boldsymbol{A}$ by $\|\boldsymbol{a}_{\cdot,k}\|_2^2$, the number of pixels in the kth spatial component. This scaling is performed so that the sizes of the spatial components do not impact when the components enter the model. That is, we would like $\|\boldsymbol{a}_{\cdot,k}\|_2^2$ to be independent of the largest value of λ for which $\hat{\boldsymbol{z}}_{k,\cdot} \neq \boldsymbol{0}$. The following lemma supports this particular scaling of the columns of $\boldsymbol{A}$.

Lemma 4.6.3 *Suppose* $\boldsymbol{Y} = \boldsymbol{A}\boldsymbol{Z}^*$ *where the following conditions hold:*

Algorithm 4 — Algorithm for Solving Equation (4.6)

1. Construct the adjacency matrix $\boldsymbol{N} \in \mathbb{R}^{K\times K}$ with $n_{i,j} = \begin{cases} 1 & \text{if } (\boldsymbol{a}_{\cdot,i})^T\boldsymbol{a}_{\cdot,j} > 0 \\ 0 & \text{if } (\boldsymbol{a}_{\cdot,i})^T\boldsymbol{a}_{\cdot,j} = 0 \end{cases}$. Let $\mathcal{N}_1, \mathcal{N}_2, \ldots, \mathcal{N}_S$ denote the connected components of the graph corresponding to $\boldsymbol{N}$. That is, $\mathcal{N}_s$ indexes the representative spatial components in the sth connected component. Define the mapping $\mathcal{M}(\mathcal{N}_s) = \{p \in (1,\ldots,P) : (\tilde{\boldsymbol{A}}_{p,\mathcal{N}_s})^T\mathbf{1} > 0\}$.

2. For $s = 1, 2, \ldots, S$, solve

$$\underset{\boldsymbol{Z}_{\mathcal{N}_s,\cdot}\geq 0}{\text{minimize}} \quad \frac{1}{2}\left\|\boldsymbol{Y}_{\mathcal{M}(\mathcal{N}_s),\cdot} - \tilde{\boldsymbol{A}}_{\mathcal{M}(\mathcal{N}_s),\mathcal{N}_s}\boldsymbol{Z}_{\mathcal{N}_s,\cdot}\right\|_F^2 + \lambda\alpha\mathbf{1}^T\boldsymbol{Z}_{\mathcal{N}_s,\cdot}\mathbf{1} + \lambda(1-\alpha)\sum_{n\in\mathcal{N}_s}\left\|\boldsymbol{z}_{n,\cdot}\right\|_2, \quad (4.9)$$

using one of the two following approaches:

   (a) By Lemma 4.6.1, if $|\mathcal{N}_s| = 1$, the closed-form solution for $\boldsymbol{z}_{\mathcal{N}_s,\cdot}$ in (4.9) is

$$\hat{\boldsymbol{z}}_{\mathcal{N}_s,\cdot} = \left(1 - \frac{\lambda(1-\alpha)}{\left\|\left((\boldsymbol{Y}_{\mathcal{M}(\mathcal{N}_s),\cdot})^T\tilde{\boldsymbol{a}}_{\mathcal{M}(\mathcal{N}_s),\mathcal{N}_s} - \lambda\alpha\mathbf{1}\right)_+\right\|_2}\right)_+ \left(\frac{(\boldsymbol{Y}_{\mathcal{M}(\mathcal{N}_s),\cdot})^T\tilde{\boldsymbol{a}}_{\mathcal{M}(\mathcal{N}_s),\mathcal{N}_s} - \lambda\alpha\mathbf{1}}{(\tilde{\boldsymbol{a}}_{\mathcal{M}(\mathcal{N}_s),\mathcal{N}_s})^T\tilde{\boldsymbol{a}}_{\mathcal{M}(\mathcal{N}_s),\mathcal{N}_s}}\right)_+. \quad (4.10)$$

   (b) If $|\mathcal{N}_s| > 1$, use generalized gradient descent to solve (4.9) for $\boldsymbol{Z}_{\mathcal{N}_s,\cdot}$:

      i. Let $f(\boldsymbol{Z}_{\mathcal{N}_s,\cdot}) = \frac{1}{2}\left\|\boldsymbol{Y}_{\mathcal{M}(\mathcal{N}_s),\cdot} - \tilde{\boldsymbol{A}}_{\mathcal{M}(\mathcal{N}_s),\mathcal{N}_s}\boldsymbol{Z}_{\mathcal{N}_s,\cdot}\right\|_F^2 + \lambda\alpha\mathbf{1}^T\boldsymbol{Z}_{\mathcal{N}_s,\cdot}\mathbf{1}$. Initialize $\boldsymbol{Z}^{(0)}_{\mathcal{N}_s,\cdot} := \mathbf{0}$ and let $t := (\max_{n\in\mathcal{N}_s}\sum_{j\in\mathcal{N}_s}(\tilde{\boldsymbol{a}}_{\mathcal{M}(\mathcal{N}_s),j})^T\tilde{\boldsymbol{a}}_{\mathcal{M}(\mathcal{N}_s),n})^{-1}$.

      ii. For $b = 1, 2, \ldots$, until convergence, iterate:

$$\nabla f\left(\boldsymbol{Z}^{(b-1)}_{\mathcal{N}_s,\cdot}\right) := -(\tilde{\boldsymbol{A}}_{\mathcal{M}(\mathcal{N}_s),\mathcal{N}_s})^T\left(\boldsymbol{Y}_{\mathcal{M}(\mathcal{N}_s),\cdot} - \tilde{\boldsymbol{A}}_{\mathcal{M}(\mathcal{N}_s),\mathcal{N}_s}\boldsymbol{Z}^{(b-1)}_{\mathcal{N}_s,\cdot}\right) + \lambda\alpha\mathbf{1}\mathbf{1}^T,$$

$$\tilde{\boldsymbol{Y}}_{\mathcal{M}(\mathcal{N}_s),\cdot} := \boldsymbol{Z}^{(b-1)}_{\mathcal{N}_s,\cdot} - t\nabla f\left(\boldsymbol{Z}^{(b-1)}_{\mathcal{N}_s,\cdot}\right), \text{ and}$$

$$\boldsymbol{z}^{(b)}_{n,\cdot} := \left(1 - \frac{\lambda(1-\alpha)t}{\left\|(\tilde{\boldsymbol{y}}_{n,\cdot})_+\right\|_2}\right)_+ (\tilde{\boldsymbol{y}}_{n,\cdot})_+ \text{ for } n \in \mathcal{N}_s.$$

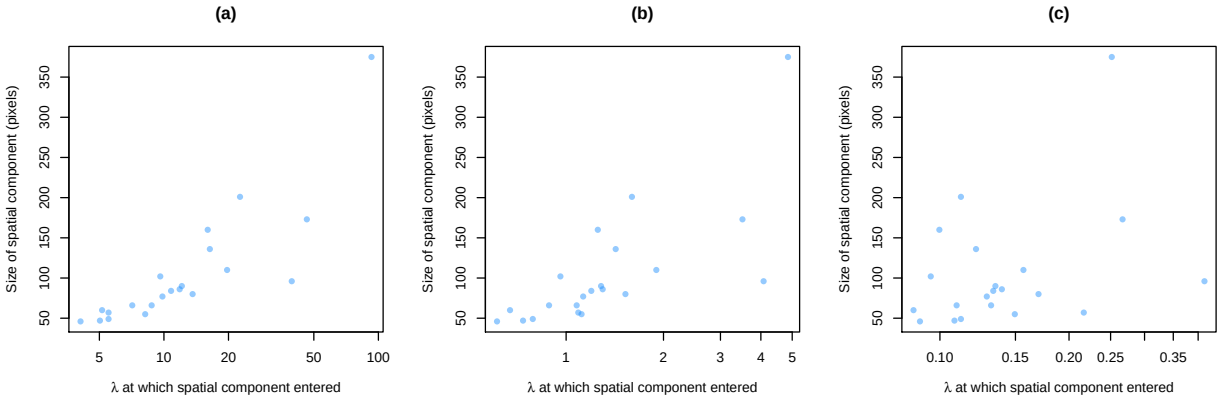


Figure 4.11: We solve (4.6) for different scalings of $\tilde{\boldsymbol{A}}$. For each spatial component, we note the value of λ at which the spatial component enters the model (i.e., the largest λ for which $\hat{\boldsymbol{z}}_{k,\cdot} \neq \boldsymbol{0}$). We plot the size of each spatial component versus the value of λ at which the spatial component enters for (a) $\tilde{\boldsymbol{a}}_{\cdot,k} = \boldsymbol{a}_{\cdot,k}$, (b) $\tilde{\boldsymbol{a}}_{\cdot,k} = \boldsymbol{a}_{\cdot,k}/\|\boldsymbol{a}_{\cdot,k}\|_2$, and (c) $\tilde{\boldsymbol{a}}_{\cdot,k} = \boldsymbol{a}_{\cdot,k}/\|\boldsymbol{a}_{\cdot,k}\|_2^2$. There is a high correlation in the scatterplots in panels (a) and (b), but little correlation in (c).

- $\boldsymbol{A} \in \mathbb{R}^{P\times K}$ *with* $(\boldsymbol{a}_{\cdot,1})^T\boldsymbol{a}_{\cdot,2} = 0$, $(\boldsymbol{a}_{\cdot,1})^T\boldsymbol{a}_{\cdot,k} = 0$ *for* $k = 3,\ldots,K$, *and* $(\boldsymbol{a}_{\cdot,2})^T\boldsymbol{a}_{\cdot,k} = 0$ *for* $k = 3,\ldots,K$ *and*

- $\boldsymbol{Z}^* \in \mathbb{R}^{K\times T}$ *with* $\boldsymbol{z}^*_{1,\cdot} = \boldsymbol{P}\boldsymbol{z}^*_{2,\cdot}$ *for some* $T \times T$ *permutation matrix* $\boldsymbol{P}$.

If we solve (4.6) *for* $\boldsymbol{Z}$ *with* $\tilde{\boldsymbol{A}}$ *such that* $\tilde{\boldsymbol{a}}_{\cdot,k} = \boldsymbol{a}_{\cdot,k}/\|\boldsymbol{a}_{\cdot,k}\|_2^2$, *then* $\hat{\boldsymbol{z}}_{1,\cdot} = \boldsymbol{0}$ *if and only if* $\hat{\boldsymbol{z}}_{2,\cdot} = \boldsymbol{0}$.

The proof of Lemma 4.6.3 is in Appendix C.8. Lemma 4.6.3 indicates that two non-overlapping spatial components, possibly of different sizes, whose temporal components are identical up to a permutation, will enter the model at the same value of λ. In Figure 4.11, we provide empirical evidence for the chosen scaling of $\boldsymbol{A}$.

4.6.4 Sparsity of the Solution

We now consider the range of λ for which the solution to (4.6) is completely sparse (i.e., $\hat{\boldsymbol{Z}} = \mathbf{0}$) for a fixed value of α.

Lemma 4.6.4 *For any $\alpha \in [0, 1]$, the solution to* (4.6) *is completely sparse if and only if*

$$\lambda(1-\alpha) \geq \left\| \left(\left[\tilde{\boldsymbol{A}}^T \boldsymbol{Y} \right]_{k,\cdot} - \lambda\alpha \mathbf{1} \right)_+ \right\|_2 \tag{4.11}$$

for $k = 1, \ldots, K$.

Unfortunately, when $\alpha \in (0, 1)$, λ is on both sides of the inequality in (4.11). Though we can solve for λ in (4.11) using a root finder when $\alpha \in (0, 1)$, the following corollary provides a simple alternative.

Corollary 4.6.5 *For any $\alpha \in (0, 1)$, if*

$$\lambda \geq \max_{k=1,\ldots,K} \left[\min \left(\max_{l=1,\ldots,T} \frac{\left(\left[\tilde{\boldsymbol{A}}^T \boldsymbol{Y} \right]_{k,l} \right)_+}{\alpha}, \frac{\left\| \left(\left[\tilde{\boldsymbol{A}}^T \boldsymbol{Y} \right]_{k,\cdot} \right)_+ \right\|_2}{1-\alpha} \right) \right],$$

then the solution to (4.6) *is completely sparse.*

The condition in Corollary 4.6.5 is sufficient, but not necessary. Proofs of Lemma 4.6.4 and Corollary 4.6.5 can be found in Appendices C.9 and C.10, respectively. An illustration of Lemma 4.6.4 and Corollary 4.6.5 is provided in Figure 4.12. In Section 4.4.3, we discuss how the results from Lemma 4.6.4 and Corollary 4.6.5 can assist in selecting λ.

4.7 Discussion

We have presented SCALPEL, a method for simultaneously identifying neurons from calcium imaging data and estimating their intracellular calcium concentrations. We use image segmentation of frames of the calcium imaging video to construct a large superset of potential

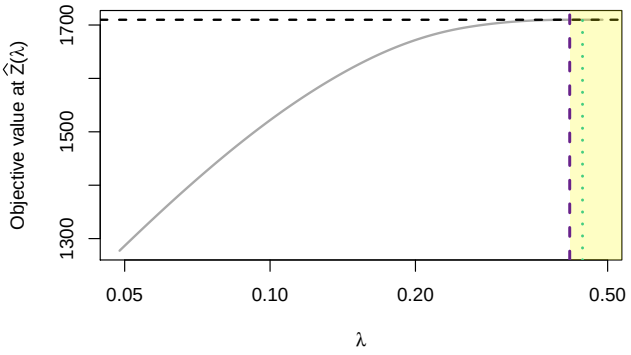


Figure 4.12: We plot the value of the objective of (4.6) at $\hat{\boldsymbol{Z}}(\lambda)$, the minimizer of (4.6) at λ, for a replicate of data as λ varies. We compare two ways of finding a λ large enough such that $\hat{\boldsymbol{Z}}(\lambda) = \mathbf{0}$, which results in the objective shown as - - -. We take λ that satisfies Lemma 4.6.4 (– –) or λ as defined in Corollary 4.6.5 (· · · ·). The former (– –) gives the smallest λ such that $\hat{\boldsymbol{Z}}(\lambda) = \mathbf{0}$, as is confirmed by being on the boundary of the shaded box that indicates the range of λ for which the objective value at $\mathbf{0}$ does not exceed the objective value at $\hat{\boldsymbol{Z}}(\lambda)$.

neurons, which are then refined through the use of clustering using a novel dissimilarity metric that leverages both spatial and temporal information. The calcium concentrations of the potential neurons are then estimated by solving a sparse group lasso with a non-negativity constraint.

Future work could consider alternative ways of deriving a dictionary of candidate spatial components in Step 1. Currently, we perform image segmentation via thresholding with a single quantile. This approach assumes that all neurons have similar brightness when spiking relative to their baseline fluorescence levels. In practice, there is evidence that some neurons, whose presence is detectable by eye, were not detected by SCALPEL due to their comparatively lower fluorescence following spiking. Additionally, while we focused on estimating the intracellular calcium concentrations over time for identified neurons, there is also interest in estimating the related spiking times of neurons. Future work could consider this.

4.8 Acknowledgements

We thank Ilana Witten for providing the calcium imaging data set that was analyzed in this chapter.

BIBLIOGRAPHY

Misha B. Ahrens, Michael B. Orger, Drew N. Robson, Jennifer M. Li, and Philipp J. Keller. Whole-brain functional imaging at cellular resolution using light-sheet microscopy. *Nature Methods*, 10(5):413–420, 2013.

Hirotugu Akaike. Information theory and an extension of the maximum likelihood principle. In *Second International Symposium on Information Theory*, pages 267–281. Akademinai Kiado, 1973.

Mark D. Alter, Rutwik Kharkar, Keri E. Ramsey, David W. Craig, Raun D. Melmed, Theresa A. Grebe, R. Curtis Bay, Sharman Ober-Reynolds, Janet Kirwan, Josh J. Jones, et al. Autism and increased paternal age related changes in global levels of gene expression regulation. *PLOS ONE*, 6(2):e16715, 2011.

Noah J. Apthorpe, Alexander J. Riordan, Rob E. Aguilar, Jan Homann, Yi Gu, David W. Tank, and H. Sebastian Seung. Automatic neuron detection in calcium imaging data using convolutional networks. *arXiv preprint arXiv:1606.07372*, 2016.

Marta Avalos, Yves Grandvalet, and Christophe Ambroise. Parsimonious additive models. *Computational Statistics & Data Analysis*, 51(6):2851–2870, 2007.

Bruno Baldessari. The distribution of a quadratic form of normal random variables. *The Annals of Mathematical Statistics*, pages 1700–1704, 1967.

Tanya Barrett, Dennis B. Troup, Stephen E. Wilhite, Pierre Ledoux, Dmitry Rudnev, Carlos Evangelista, Irene F. Kim, Alexandra Soboleva, Maxim Tomashevsky, and Ron Edgar. NCBI GEO: Mining tens of millions of expression profiles — database and tools update. *Nucleic Acids Research*, 35(suppl. 1):D760–D765, 2007.

Amir Beck and Marc Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM Journal on Imaging Sciences*, 2(1):183–202, 2009.

Jacob Bien and Rob Tibshirani. *protoclust: Hierarchical Clustering with Prototypes*, 2015. URL `https://CRAN.R-project.org/package=protoclust`. R package version 1.5.

Jacob Bien and Robert Tibshirani. Hierarchical clustering with prototypes via minimax linkage. *Journal of the American Statistical Association*, 106(495):1075–1084, 2011.

Stephen Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge University Press, 2004.

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.

Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

Leo Breiman, Jerome Friedman, Charles J. Stone, and Richard A. Olshen. *Classification and Regression Trees*. CRC Press, 1984.

Lars Buesing, Timothy A. Machado, John P. Cunningham, and Liam Paninski. Clustered factor analysis of multineuronal spike data. In *Advances in Neural Information Processing Systems*, pages 3500–3508, 2014.

Peter Bühlmann and Sara van de Geer. *Statistics for High-Dimensional Data: Methods, Theory and Applications*. Springer, 2011.

Jian-Feng Cai, Emmanuel J. Candès, and Zuowei Shen. A singular value thresholding algorithm for matrix completion. *SIAM Journal on Optimization*, 20(4):1956–1982, 2010.

Emmanuel J. Candès and Yaniv Plan. Matrix completion with noise. *Proceedings of the IEEE*, 98(6):925–936, 2010.

Emmanuel J. Candès and Benjamin Recht. Exact matrix completion via convex optimization. *Foundations of Computational Mathematics*, 9(6):717–772, 2009.

Emmanuel J. Candès and Terence Tao. The power of convex relaxation: Near-optimal matrix completion. *IEEE Transactions on Information Theory*, 56(5):2053–2080, 2010.

Hadley Cantril. Pattern of human concerns. 1965.

Tsai-Wen Chen, Trevor J. Wardill, Yi Sun, Stefan R. Pulver, Sabine L. Renninger, Amy Baohan, Eric R. Schreiter, Rex A. Kerr, Michael B. Orger, Vivek Jayaraman, et al. Ultrasensitive fluorescent proteins for imaging neuronal activity. *Nature*, 499(7458):295–300, 2013.

Eric C. Chi, Genevera I. Allen, and Richard G. Baraniuk. Convex biclustering. *Biometrics*, forthcoming.

Carl de Boor. A practical guide to splines. *Mathematics of Computation*, 1978.

Ferran Diego, Susanne Reichinnek, Martin Both, Fred Hamprecht, et al. Automated identification of neuronal activity from calcium imaging by sparse dictionary learning. In *Biomedical Imaging (ISBI), 2013 IEEE 10th International Symposium on*, pages 1058–1061. IEEE, 2013.

Ferran Diego Andilla and Fred A. Hamprecht. Learning multi-level sparse representations. In *Advances in Neural Information Processing Systems*, pages 818–826, 2013.

Ferran Diego Andilla and Fred A. Hamprecht. Sparse space-time deconvolution for calcium image analysis. In *Advances in Neural Information Processing Systems*, pages 64–72, 2014.

Daniel A. Dombeck, Anton N. Khabbaz, Forrest Collman, Thomas L. Adelman, and David W. Tank. Imaging large-scale neural activity with cellular resolution in awake, mobile mice. *Neuron*, 56(1):43–57, 2007.

John Duchi and Yoram Singer. Efficient online and batch learning using forward backward splitting. *The Journal of Machine Learning Research*, 10:2899–2934, 2009.

Jean Duchon. Splines minimizing rotation-invariant semi-norms in Sobolev spaces. In *Constructive Theory of Functions of Several Variables*, pages 85–100. Springer, 1977.

Bradley Efron. How biased is the apparent error rate of a prediction rule? *Journal of the American Statistical Association*, 81(394):461–470, 1986.

Maryam Fazel. Matrix rank minimization with applications. *Elec. Eng. Dept. Stanford University*, 54:1–130, 2002.

Alyson K. Fletcher and Sundeep Rangan. Scalable inference for neuronal connectivity from calcium imaging. In *Advances in Neural Information Processing Systems*, pages 2843–2851, 2014.

Jerome H. Friedman. Multivariate adaptive regression splines. *The Annals of Statistics*, pages 1–67, 1991.

Johannes Friedrich, Daniel Soudry, Yu Mu, Jeremy Freeman, M. Ahres, and Liam Paninski. Fast constrained non-negative matrix factorization for whole-brain calcium imaging data. In *NIPS Workshop on Statistical Methods for Understanding Neural Systems*, 2015.

J.C. Gower. Similarity, dissimilarity and distance, measures of. *Encyclopedia of Statistical Sciences*, 2006.

Michael Grant and Stephen Boyd. Graph implementations for nonsmooth convex programs. In V. Blondel, S. Boyd, and H. Kimura, editors, *Recent Advances in Learning and Control*, Lecture Notes in Control and Information Sciences, pages 95–110. Springer-Verlag Limited, 2008. `http://stanford.edu/~boyd/graph_dcp.html`.

Michael Grant and Stephen Boyd. CVX: Matlab software for disciplined convex programming, version 2.1. `http://cvxr.com/cvx`, March 2014.

Benjamin F. Grewe, Dominik Langer, Hansjörg Kasper, Björn M. Kampa, and Fritjof Helmchen. High-speed in vivo calcium imaging reveals neuronal network activity with near-millisecond precision. *Nature Methods*, 7(5):399–405, 2010.

Christine Grienberger and Arthur Konnerth. Imaging calcium in neurons. *Neuron*, 73(5): 862–885, 2012.

Benjamin Haeffele, Eric Young, and Rene Vidal. Structured low-rank matrix factorization: Optimality, algorithm, and applications to image processing. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pages 2007–2015, 2014.

Trevor Hastie. *gam: Generalized Additive Models*, 2013. URL `http://CRAN.R-project.org/package=gam`. R package version 1.09.

Trevor Hastie and Robert Tibshirani. Generalized additive models. *Statistical Science*, pages 297–310, 1986.

Trevor Hastie, Robert Tibshirani, and Jerome H. Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer, 2009.

Trevor J. Hastie and Robert J. Tibshirani. *Generalized Additive Models*, volume 43. CRC Press, 1990.

John F. Helliwell, Richard Layard, Jeffrey Sachs, and Emirates Competitiveness Council. *World happiness report 2013*. Sustainable Development Solutions Network, 2013.

Fritjof Helmchen and Winfried Denk. Deep tissue two-photon microscopy. *Nature Methods*, 2(12):932–940, 2005.

Robert J. Hodrick and Edward C. Prescott. Postwar us business cycles: an empirical investigation. *Journal of Money, Credit, and Banking*, pages 1–16, 1997.

Holger Hoefling. A path algorithm for the fused lasso signal approximator. *Journal of Computational and Graphical Statistics*, 19(4):984–1006, 2010.

Holger Hoefling. *flsa: Path algorithm for the general Fused Lasso Signal Approximator*, 2013. URL `http://CRAN.R-project.org/package=flsa`. R package version 1.05.

Arthur E. Hoerl and Robert W. Kennard. Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, 12(1):55–67, 1970.

Jian Huang, Joel L. Horowitz, and Fengrong Wei. Variable selection in nonparametric additive models. *The Annals of Statistics*, 38(4):2282, 2010.

Daniel Huber, D.A. Gutnisky, S. Peron, D.H. Oconnor, J.S. Wiegert, Lin Tian, T.G. Oertner, L.L. Looger, and K. Svoboda. Multiple dynamic representations in the motor cortex during sensorimotor learning. *Nature*, 484(7395):473–478, 2012.

Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani. *An Introduction to Statistical Learning*. Springer, 2013.

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.

Raghunandan Keshavan, Andrea Montanari, and Sewoong Oh. Matrix completion from noisy entries. In *Advances in Neural Information Processing Systems*, pages 952–960, 2009.

Raghunandan H. Keshavan, Andrea Montanari, and Sewoong Oh. Matrix completion from a few entries. *IEEE Transactions on Information Theory*, 56(6):2980–2998, 2010.

Seung-Jean Kim, Kwangmoo Koh, Stephen Boyd, and Dimitry Gorinevsky. ℓ_1 trend filtering. *SIAM Review*, 51(2):339–360, 2009.

Vladimir Koltchinskii, Karim Lounici, and Alexandre B. Tsybakov. Nuclear-norm penalization and optimal rates for noisy low-rank matrix completion. *The Annals of Statistics*, 39 (5):2302–2329, 2011.

Beatrice Laurent and Pascal Massart. Adaptive estimation of a quadratic functional by model selection. *The Annals of Statistics*, pages 1302–1338, 2000.

Yan Li and Han Liu. Sparse additive model using symmetric nonnegative definite smoothers. *arXiv preprint arXiv:1409.2552*, 2014.

Yi Lin and Hao Helen Zhang. Component selection and smoothing in smoothing spline analysis of variance models. *The Annals of Statistics*, 34(5):2272–2297, 2006.

Jun Liu, Lei Yuan, and Jieping Ye. An efficient algorithm for a class of fused lasso problems. In *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 323–332. ACM, 2010.

Loren L. Looger and Oliver Griesbeck. Genetically encoded neural activity indicators. *Current Opinion in Neurobiology*, 22(1):18–23, 2012.

Yin Lou, Jacob Bien, Rich Caruana, and Johannes Gehrke. Sparse partially linear additive models. *Journal of Computational and Graphical Statistics*, forthcoming.

Tzu-Pin Lu, Mong-Hsun Tsai, Jang-Ming Lee, Chung-Ping Hsu, Pei-Chun Chen, Chung-Wu Lin, Jin-Yuan Shih, Pan-Chyr Yang, Chuhsing Kate Hsiao, Liang-Chuan Lai, et al. Identification of a novel biomarker, SEMA5A, for non-small cell lung carcinoma in nonsmoking women. *Cancer Epidemiology Biomarkers & Prevention*, 19(10):2590–2597, 2010.

Henry Lütcke, Felipe Gerhard, Friedemann Zenke, Wulfram Gerstner, and Fritjof Helmchen. Inference of neuronal network spike dynamics and topology from calcium imaging data. *Frontiers in neural circuits*, 7, 2013.

Enno Mammen and Sara van de Geer. Locally adaptive regression splines. *The Annals of Statistics*, 25(1):387–413, 1997.

Ryuichi Maruyama, Kazuma Maeda, Hajime Moroda, Ichiro Kato, Masashi Inoue, Hiroyoshi Miyakawa, and Toru Aonishi. Detecting cells using non-negative matrix factorization on calcium imaging data. *Neural Networks*, 55:11–19, 2014.

Rahul Mazumder, Trevor Hastie, and Robert Tibshirani. Spectral regularization algorithms for learning large incomplete matrices. *The Journal of Machine Learning Research*, 11: 2287–2322, 2010.

Lukas Meier, Sara van de Geer, and Peter Bühlmann. High-dimensional additive modeling. *The Annals of Statistics*, 37(6B):3779–3821, 2009.

Eran A. Mukamel, Axel Nimmerjahn, and Mark J. Schnitzer. Automated analysis of cellular signals from large-scale calcium imaging data. *Neuron*, 63(6):747–760, 2009.

Sahand Negahban and Martin J. Wainwright. Restricted strong convexity and weighted matrix completion: Optimal bounds with noise. *The Journal of Machine Learning Research*, 13(1):1665–1697, 2012.

Douglas Nychka, Reinhard Furrer, and Stephan Sain. *fields: Tools for spatial data*, 2014. URL `http://CRAN.R-project.org/package=fields`. R package version 7.1.

Jon Onativia, Simon R. Schultz, and Pier Luigi Dragotti. A finite rate of innovation algorithm for fast and accurate spike detection from two-photon calcium imaging. *Journal of neural engineering*, 10(4):046017, 2013.

R. Kelley Pace and Ronald Barry. Sparse spatial autoregressions. *Statistics & Probability Letters*, 33(3):291–297, 1997.

Marius Pachitariu, Adam M. Packer, Noah Pettit, Henry Dalgleish, Michael Hausser, and Maneesh Sahani. Extracting regions of interest from biological images with convolutional sparse block coding. In *Advances in Neural Information Processing Systems*, pages 1745–1753, 2013.

Neal Parikh and Stephen Boyd. Proximal algorithms. *Foundations and Trends in Optimization*, 1(3):127–239, 2014.

Sven Peter, Daniel Durstewitz, Ferran Diego, and Fred A. Hamprecht. Sparse convolutional coding for neuronal ensemble identification. *arXiv preprint arXiv:1606.07029*, 2016.

Ashley Petersen. *flam: Fits Piecewise Constant Models with Data-Adaptive Knots*, 2014. URL `http://CRAN.R-project.org/package=flam`. R package version 1.0.

Ashley Petersen, Noah Simon, and Daniela Witten. Convex regression with interpretable sharp partitions. *Journal of Machine Learning Research*, 17(94):1–31, 2016.

Ashley Petersen, Daniela Witten, and Noah Simon. Fused lasso additive model. *Journal of Computational and Graphical Statistics*, forthcoming.

Jonathan W. Pillow, Yashar Ahmadian, and Liam Paninski. Model-based decoding, information estimation, and change-point detection techniques for multineuron spike trains. *Neural Computation*, 23(1):1–45, 2011.

Eftychios Pnevmatikakis, Josh Merel, Ari Pakman, Liam Paninski, et al. Bayesian spike inference from calcium imaging data. In *Signals, Systems and Computers, 2013 Asilomar Conference on*, pages 349–353. IEEE, 2013a.

Eftychios A. Pnevmatikakis and Liam Paninski. Sparse nonnegative deconvolution for compressive calcium imaging: algorithms and phase transitions. In *Advances in Neural Information Processing Systems*, pages 1250–1258, 2013.

Eftychios A. Pnevmatikakis, Timothy A. Machado, Logan Grosenick, Ben Poole, Joshua T. Vogelstein, and Liam Paninski. Rank-penalized nonnegative spatiotemporal deconvolution and demixing of calcium imaging data. In *Computational and Systems Neuroscience Meeting COSYNE*, 2013b.

Eftychios A. Pnevmatikakis, Yuanjun Gao, Daniel Soudry, David Pfau, Clay Lacefield, Kira Poskanzer, Randy Bruno, Rafael Yuste, and Liam Paninski. A structured matrix factorization framework for large scale calcium imaging data analysis. *arXiv preprint arXiv:1409.2903*, 2014.

Eftychios A. Pnevmatikakis, Daniel Soudry, Yuanjun Gao, Timothy A. Machado, Josh Merel, David Pfau, Thomas Reardon, Yu Mu, Clay Lacefield, Weijian Yang, et al. Simultaneous denoising, deconvolution, and demixing of calcium imaging data. *Neuron*, 89(2):285–299, 2016.

Robert Prevedel, Young-Gyu Yoon, Maximilian Hoffmann, Nikita Pak, Gordon Wetzstein, Saul Kato, Tina Schrödel, Ramesh Raskar, Manuel Zimmer, Edward S Boyden, et al. Simultaneous whole-animal 3d imaging of neuronal activity using light-field microscopy. *Nature Methods*, 11(7):727–730, 2014.

R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2013. URL `http://www.R-project.org/`.

Aaditya Ramdas and Ryan J. Tibshirani. Fast and flexible ADMM algorithms for trend filtering. *Journal of Computational and Graphical Statistics*, forthcoming.

Pradeep Ravikumar, John Lafferty, Han Liu, and Larry Wasserman. Sparse additive models. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 71(5):1009–1030, 2009.

Benjamin Recht. A simpler approach to matrix completion. *The Journal of Machine Learning Research*, 12:3413–3430, 2011.

Nathalie L. Rochefort, Hongbo Jia, and Arthur Konnerth. Calcium imaging in the living brain: prospects for molecular medicine. *Trends in Molecular Medicine*, 14(9):389–399, 2008.

RStudio and Inc. *shiny: Web Application Framework for R*, 2014. URL `http://CRAN.R-project.org/package=shiny`. R package version 0.9.1.

Sylvain Sardy and Paul Tseng. AMlet, RAMlet, and GAMlet: Automatic nonlinear fitting of additive models, robust and generalized, with wavelets. *Journal of Computational and Graphical Statistics*, 13(2):283–309, 2004.

Gideon Schwarz. Estimating the dimension of a model. *The Annals of Statistics*, 6(2): 461–464, 1978.

Noah Simon and Robert Tibshirani. Standardization and the group lasso penalty. *Statistica Sinica*, 22(3):983, 2012.

Noah Simon, Jerome Friedman, Trevor Hastie, and Robert Tibshirani. A sparse-group lasso. *Journal of Computational and Graphical Statistics*, 22(2):231–245, 2013.

Spencer L. Smith and Michael Häusser. Parallel processing of visual space by neighboring neurons in mouse visual cortex. *Nature neuroscience*, 13(9):1144–1149, 2010.

Avrum Spira, Jennifer E. Beane, Vishal Shah, Katrina Steiling, Gang Liu, Frank Schembri, Sean Gilman, Yves-Martine Dumas, Paul Calner, Paola Sebastiani, et al. Airway epithelial gene expression in the diagnostic evaluation of smokers with suspect lung cancer. *Nature Medicine*, 13(3):361–366, 2007.

Karel Svoboda and Ryohei Yasuda. Principles of two-photon excitation microscopy and its applications to neuroscience. *Neuron*, 50(6):823–839, 2006.

Terry Therneau, Beth Atkinson, and Brian Ripley. *rpart: Recursive Partitioning and Regression Trees*, 2014. URL `http://CRAN.R-project.org/package=rpart`. R package version 4.1-8.

Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, pages 267–288, 1996.

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 (Statistical Methodology)*, 67(1):91–108, 2005.

Ryan J. Tibshirani. Adaptive piecewise polynomial estimation via trend filtering. *The Annals of Statistics*, 42(1):285–323, 02 2014.

Ryan J. Tibshirani and Jonathan Taylor. Degrees of freedom in lasso problems. *The Annals of Statistics*, 40(2):1198–1232, 2012.

Jakub Tomek, Ondrej Novak, and Josef Syka. Two-photon processor and seneca: a freely available software package to process data from two-photon calcium imaging at speeds down to several milliseconds per frame. *Journal of neurophysiology*, 110(1):243–256, 2013.

Paul Tseng. Convergence of a block coordinate descent method for nondifferentiable minimization. *Journal of Optimization Theory and Applications*, 109(3):475–494, 2001.

UNDP. *Human Development Indicators*. United Nations Publications, 2012.

Sara van de Geer. *Empirical Processes in M-estimation*, volume 6. Cambridge University Press, 2000.

Joshua T. Vogelstein, Brendon O. Watson, Adam M. Packer, Rafael Yuste, Bruno Jedynak, and Liam Paninski. Spike inference from calcium imaging using sequential monte carlo methods. *Biophysical journal*, 97(2):636–655, 2009.

Joshua T. Vogelstein, Adam M. Packer, Timothy A. Machado, Tanya Sippy, Baktash Babadi, Rafael Yuste, and Liam Paninski. Fast nonnegative deconvolution for spike train inference from population calcium imaging. *Journal of neurophysiology*, 104(6):3691–3704, 2010.

Simon N. Wood, Yannig Goude, and Simon Shaw. Generalized additive models for large data sets. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, 2014.

World Bank Group. *World Development Indicators 2012*. World Bank Publications, 2012.

Jianming Ye. On measuring and correcting the effects of data mining and model selection. *Journal of the American Statistical Association*, 93(441):120–131, 1998.

Ming Yuan and Yi Lin. Model selection and estimation in regression with grouped variables. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 68(1):49–67, 2006.

Hao Helen Zhang, Guang Cheng, and Yufeng Liu. Linear or nonlinear? Automatic structure discovery for partially linear models. *Journal of the American Statistical Association*, 106 (495), 2011.

Tuo Zhao, Xingguo Li, Han Liu, and Kathryn Roeder. *SAM: Sparse Additive Modelling*, 2014. URL `http://CRAN.R-project.org/package=SAM`. R package version 1.0.5.

Pengcheng Zhou, Shanna L. Resendez, Garret D. Stuber, Robert E. Kass, and Liam Paninski. Efficient and accurate extraction of in vivo calcium signals from microendoscopic video data. *arXiv preprint arXiv:1605.07266*, 2016.

Hui Zou, Trevor Hastie, and Robert Tibshirani. On the "degrees of freedom" of the lasso. *The Annals of Statistics*, 35(5):2173–2192, 2007.

Appendix A

APPENDIX FOR CHAPTER 2

A.1 Proof of Lemma 2.3.1

Proof Differentiating (2.5) with respect to θ_0 gives $\hat{\theta}_0 = \frac{1}{n}\mathbf{1}^T\left(\boldsymbol{y} - \sum_{j=1}^p \boldsymbol{\theta}_j\right) = \frac{1}{n}\mathbf{1}^T\boldsymbol{y}$, since $\mathbf{1}^T\boldsymbol{\theta}_j = 0$ for $j = 1, \ldots, p$. Thus we can solve (2.5) with $\boldsymbol{y}$ centered and no intercept. We now reparameterize in terms of $\boldsymbol{\beta}_j \in \mathbb{R}^{n-1}$ where $\boldsymbol{\beta}_j = \boldsymbol{D}\boldsymbol{P}_j\boldsymbol{\theta}_j$. Note that

$$\boldsymbol{\theta}_j = \boldsymbol{P}_j^T\boldsymbol{P}_j\boldsymbol{\theta}_j = \boldsymbol{P}_j^T\left(\boldsymbol{U}\boldsymbol{D} + \frac{1}{n}\mathbf{1}\mathbf{1}^T\right)\boldsymbol{P}_j\boldsymbol{\theta}_j = \boldsymbol{P}_j^T\boldsymbol{U}\boldsymbol{\beta}_j,$$

using the facts that $\boldsymbol{U}\boldsymbol{D} + \frac{1}{n}\mathbf{1}\mathbf{1}^T = \boldsymbol{I}$ and $\mathbf{1}^T\boldsymbol{P}_j\boldsymbol{\theta}_j = 0$. Defining $\boldsymbol{\beta} = (\boldsymbol{\beta}_1^T \ \cdots \ \boldsymbol{\beta}_p^T)^T \in \mathbb{R}^{(n-1)p}$ and $\boldsymbol{V} = [\boldsymbol{P}_1^T\boldsymbol{U} \cdots \boldsymbol{P}_p^T\boldsymbol{U}]$, we see that (2.5) can be rewritten as (2.8). ■

A.2 Proof of Proposition 2.3.2

Proof We first derive the degrees of freedom for $\hat{\boldsymbol{y}}$ for (2.10) when $\boldsymbol{y} \sim MVN(\mathbf{0}, \sigma^2\boldsymbol{I})$. Using the dual problem of (2.10) and Lemma 1 of Tibshirani and Taylor [2012], it can be shown that $g : \mathbb{R}^n \to \mathbb{R}^n$ with $\hat{\boldsymbol{y}} = g(\boldsymbol{y}) = (g_1(\boldsymbol{y}), \ldots, g_n(\boldsymbol{y}))^T$ is continuous and almost differentiable. Thus, Stein's lemma implies that $\mathrm{df}(\hat{\boldsymbol{y}}) = \mathrm{E}\left[\mathrm{Tr}\left(\frac{\partial g(\boldsymbol{y})}{\partial \boldsymbol{y}}\right)\right]$. We denote the active set of $\hat{\boldsymbol{\beta}}$ as $\mathcal{A}$, which is unique since (2.10) is strictly convex. At the optimum of (2.10), we have

$$\mathbf{0} = -\boldsymbol{V}_{\mathcal{A}}^T\left(\boldsymbol{y} - \boldsymbol{V}_{\mathcal{A}}\hat{\boldsymbol{\beta}}_{\mathcal{A}}\right) + \alpha\lambda\mathrm{sign}\left(\hat{\boldsymbol{\beta}}_{\mathcal{A}}\right) + (1-\alpha)\lambda S_1 + \gamma\hat{\boldsymbol{\beta}}_{\mathcal{A}}, \tag{A.1}$$

where $S_1 = \frac{\partial}{\partial \hat{\boldsymbol{\beta}}_{\mathcal{A}}}\sum_{j=1}^p \|\boldsymbol{U}\boldsymbol{\beta}_j\|_2 = \left[S_{1,1}^T \cdots S_{1,p}^T\right]^T$ with $S_{1,j} = \frac{\boldsymbol{U}_{\mathcal{A}_j}^T\boldsymbol{U}_{\mathcal{A}_j}\hat{\boldsymbol{\beta}}_{j\mathcal{A}_j}}{\|\boldsymbol{U}_{\mathcal{A}_j}\hat{\boldsymbol{\beta}}_{j\mathcal{A}_j}\|_2}$.

We conjecture that there is a neighborhood around almost every $\boldsymbol{y}$ (i.e., except a set of measure zero) such that $\hat{\boldsymbol{\beta}}^*$ corresponding to any $\boldsymbol{y}^*$ in that neighborhood has $\mathcal{A}^* = \mathcal{A}$ and

$\text{sign}(\hat{\boldsymbol{\beta}}^*_{\mathcal{A}^*}) = \text{sign}(\hat{\boldsymbol{\beta}}_{\mathcal{A}})$. On the basis of this conjecture, we treat $\mathcal{A}$ and $\text{sign}(\hat{\boldsymbol{\beta}}_{\mathcal{A}})$ in (A.1) as constants with respect to $\boldsymbol{y}$. Thus the derivative of (A.1) with respect to $\boldsymbol{y}$ is

$$\mathbf{0} = -\boldsymbol{V}_{\mathcal{A}}^T + \boldsymbol{V}_{\mathcal{A}}^T \boldsymbol{V}_{\mathcal{A}} \frac{\partial \hat{\boldsymbol{\beta}}_{\mathcal{A}}}{\partial \boldsymbol{y}} + (1-\alpha)\lambda S_2 \frac{\partial \hat{\boldsymbol{\beta}}_{\mathcal{A}}}{\partial \boldsymbol{y}} + \gamma \frac{\partial \hat{\boldsymbol{\beta}}_{\mathcal{A}}}{\partial \boldsymbol{y}}, \tag{A.2}$$

where $S_2 = \frac{\partial S_1}{\partial \hat{\boldsymbol{\beta}}_{\mathcal{A}}}$ is a block diagonal matrix with the jth block equaling

$$\frac{\boldsymbol{U}_{\mathcal{A}_j}^T \boldsymbol{U}_{\mathcal{A}_j}}{\|\boldsymbol{U}_{\mathcal{A}_j} \hat{\boldsymbol{\beta}}_{j\mathcal{A}_j}\|_2} - \frac{\boldsymbol{U}_{\mathcal{A}_j}^T \boldsymbol{U}_{\mathcal{A}_j} \hat{\boldsymbol{\beta}}_{j\mathcal{A}_j} \hat{\boldsymbol{\beta}}_{j\mathcal{A}_j}^T \boldsymbol{U}_{\mathcal{A}_j}^T \boldsymbol{U}_{\mathcal{A}_j}}{\|\boldsymbol{U}_{\mathcal{A}_j} \hat{\boldsymbol{\beta}}_{j\mathcal{A}_j}\|_2^3}.$$

Solving (A.2) for $\frac{\partial \hat{\boldsymbol{\beta}}_{\mathcal{A}}}{\partial \boldsymbol{y}}$ and left multiplying by $\boldsymbol{V}_{\mathcal{A}}$, we have

$$\frac{\partial \hat{\boldsymbol{y}}}{\partial \boldsymbol{y}} = \boldsymbol{V}_{\mathcal{A}} \frac{\partial \hat{\boldsymbol{\beta}}_{\mathcal{A}}}{\partial \boldsymbol{y}} = \boldsymbol{V}_{\mathcal{A}} \left[\boldsymbol{V}_{\mathcal{A}}^T \boldsymbol{V}_{\mathcal{A}} + (1-\alpha)\lambda S_2 + \gamma \boldsymbol{I} \right]^{-1} \boldsymbol{V}_{\mathcal{A}}^T.$$

Therefore, the degrees of freedom are $\text{E}\left[\text{Tr}\left(\boldsymbol{V}_{\mathcal{A}} \left[\boldsymbol{V}_{\mathcal{A}}^T \boldsymbol{V}_{\mathcal{A}} + (1-\alpha)\lambda S_2 + \gamma \boldsymbol{I} \right]^{-1} \boldsymbol{V}_{\mathcal{A}}^T \right)\right]$. This yields the estimator (2.11), where one degree of freedom is added for the intercept. ■

The proof of Proposition 2.3.4 is omitted, as it follows the arguments in this proof closely.

A.3 Proof of Lemma 2.3.5

Proof The optimality condition for (2.8) with $\alpha = 1$ is $-\boldsymbol{V}^T(\tilde{\boldsymbol{y}} - \boldsymbol{V}\boldsymbol{\beta}) + \lambda s(\boldsymbol{\beta}) = 0$, where $s(\boldsymbol{\beta}_j) = \text{sign}(\boldsymbol{\beta}_j)$ if $\boldsymbol{\beta}_j \neq 0$ and $s(\boldsymbol{\beta}_j) \in [-1, 1]$ if $\boldsymbol{\beta}_j = 0$. After plugging in $\boldsymbol{\beta} = \mathbf{0}$, we obtain $-\boldsymbol{V}^T \tilde{\boldsymbol{y}} + \lambda s(\mathbf{0}) = \mathbf{0}$, which is satisfied if and only if $\lambda \geq \|\boldsymbol{V}^T \tilde{\boldsymbol{y}}\|_\infty$ since $s(\mathbf{0}) \in [-1, 1]$.

Now we consider the optimality condition for (2.5) when $\alpha = 0$, which takes the form $-\tilde{\boldsymbol{y}} + \sum_{j=1}^p \boldsymbol{\theta}_j + \lambda s(\boldsymbol{\theta}_j) = \mathbf{0}$ for $j = 1, \ldots, p$, where $s(\boldsymbol{\theta}_j) = \boldsymbol{\theta}_j / \|\boldsymbol{\theta}_j\|_2$ if $\boldsymbol{\theta}_j \neq \mathbf{0}$ and $s(\boldsymbol{\theta}_j) \in \{\boldsymbol{g} \mid \|\boldsymbol{g}\|_2 \leq 1\}$ if $\boldsymbol{\theta}_j = \mathbf{0}$. After plugging in $\boldsymbol{\theta}_j = \mathbf{0}$ for $j = 1, \ldots, p$, we obtain $-\tilde{\boldsymbol{y}} + \lambda s(\mathbf{0}) = \mathbf{0}$, which is satisfied if and only if $\lambda \geq \|\tilde{\boldsymbol{y}}\|_2$. ■

A.4 Proof of Corollary 2.3.6

Proof The objective of (2.5) is bounded below by

$$\min_{\theta_0\in\mathbb{R},\boldsymbol{\theta}_j\in\mathbb{R}^n,1\le j\le p} \quad \frac{1}{2}\left\|\boldsymbol{y}-\sum_{j=1}^{p}\boldsymbol{\theta}_j-\theta_0\mathbf{1}\right\|_2^2+\alpha\lambda\sum_{j=1}^{p}\|\boldsymbol{D}\boldsymbol{P}_j\boldsymbol{\theta}_j\|_1 \tag{A.3}$$

and

$$\min_{\theta_0\in\mathbb{R},\boldsymbol{\theta}_j\in\mathbb{R}^n,1\le j\le p} \quad \frac{1}{2}\left\|\boldsymbol{y}-\sum_{j=1}^{p}\boldsymbol{\theta}_j-\theta_0\mathbf{1}\right\|_2^2+(1-\alpha)\lambda\sum_{j=1}^{p}\|\boldsymbol{\theta}_j\|_2 \tag{A.4}$$

for any $\alpha\in[0,1]$ and $\lambda\ge 0$. Lemma 2.3.5 implies that if $\lambda\ge\min\left(\frac{\|\boldsymbol{V}^T\tilde{\boldsymbol{y}}\|_\infty}{\alpha},\frac{\|\tilde{\boldsymbol{y}}\|_2}{1-\alpha}\right)$, then the objective of (2.5) is bounded below by $\min_{\theta_0\in\mathbb{R}}\ \frac{1}{2}\|\boldsymbol{y}-\theta_0\mathbf{1}\|_2^2=\frac{1}{2}\left\|\boldsymbol{y}-(\frac{1}{n}\mathbf{1}^T\boldsymbol{y})\mathbf{1}\right\|_2^2$. The objective of (2.5) achieves this lower bound when $\hat{\boldsymbol{\theta}}_j=\mathbf{0}$ for $j=1,\ldots,p$. ■

A.5 Proof of Lemma 2.3.7

Fact A.5.1 *Let $\boldsymbol{z}_j\in\mathbb{R}^n\sim MVN(\mathbf{0},\Sigma)$ with $\max_{1\le i\le n}\Sigma_{i,i}\le c$ for $j=1,\ldots,p$. Then*

$$P\left(\max_{1\le j\le p}\|\boldsymbol{z}_j\|_\infty\ge 2\sqrt{c\log(np)}\right)\le\frac{2}{np}.$$

Proof Note that $P(z_{ji}\ge 2\sqrt{c\log(np)})\le P(z_0\ge 2\sqrt{c\log(np)})$ where $z_0\sim N(0,c)$. Thus

$$P\left(\max_{1\le j\le p}\|\boldsymbol{z}_j\|_\infty\ge 2\sqrt{c\log(np)}\right)\le 2npP\left(z_0/\sqrt{c}\ge 2\sqrt{\log(np)}\right)\le\frac{2}{np},$$

which follows from the union bound and the fact that

$$P\left(z_0/\sqrt{c}\ge 2\sqrt{\log(np)}\right)=\int_{2\sqrt{\log(np)}}^{\infty}\frac{1}{\sqrt{2\pi}}\exp\left(\frac{-t^2}{2}\right)dt\le\int_{2\sqrt{\log(np)}}^{\infty}t\exp\left(\frac{-t^2}{2}\right)dt=\frac{1}{n^2p^2}.$$

■

Fact A.5.2 *Let $\boldsymbol{z}\in\mathbb{R}^n\sim MVN(\mathbf{0},c\boldsymbol{I})$. Then $P\left(\|\boldsymbol{z}\|_2\ge\sqrt{cn\left(1+\sqrt{\frac{4\log n}{n}}+\frac{4\log n}{n}\right)}\right)\le\frac{1}{n}$.*

Proof Since $\|\boldsymbol{z}\|_2^2 \sim c\chi_n^2$, this follows from Lemma 8.1 in Bühlmann and van de Geer [2011]. ■

Proof We now establish prediction consistency. We rewrite (2.5) as

$$\underset{\boldsymbol{\theta}\in\mathbb{R}^{np}}{\text{minimize}} \quad \frac{1}{2n}\|\boldsymbol{y}-\boldsymbol{W}\boldsymbol{\theta}\|_2^2 + \lambda\Omega(\boldsymbol{\theta}),$$

where $\boldsymbol{W} = \mathbf{1}^T \otimes \boldsymbol{I}_n$ and $\Omega(\boldsymbol{\theta}) = \sum_{j=1}^p \left[\alpha \|\boldsymbol{D}\boldsymbol{P}_j\boldsymbol{\theta}_j\|_1 + (1-\alpha)\|\boldsymbol{\theta}_j\|_2\right]$. Denote the true coefficient vector as $\boldsymbol{\theta}^0 = (\boldsymbol{\theta}_1^{0T}\cdots\boldsymbol{\theta}_p^{0T})^T$ and assume $\boldsymbol{y} = \boldsymbol{W}\boldsymbol{\theta}^0 + \boldsymbol{\epsilon}$ with $\boldsymbol{\epsilon} \sim MVN(\mathbf{0}, \sigma^2\boldsymbol{I})$. By the definition of $\hat{\boldsymbol{\theta}}$, $\frac{1}{2n}\left\|\boldsymbol{y}-\boldsymbol{W}\hat{\boldsymbol{\theta}}\right\|_2^2 + \lambda\Omega(\hat{\boldsymbol{\theta}}) \le \frac{1}{2n}\|\boldsymbol{y}-\boldsymbol{W}\boldsymbol{\theta}^0\|_2^2 + \lambda\Omega(\boldsymbol{\theta}^0)$, so

$$\frac{1}{2n}\left\|\sum_{j=1}^p(\hat{\boldsymbol{\theta}}_j - \boldsymbol{\theta}_j^0)\right\|_2^2 + \lambda\Omega(\hat{\boldsymbol{\theta}}) \le V_n(\hat{\boldsymbol{\theta}}) + \lambda\Omega(\boldsymbol{\theta}^0) \tag{A.5}$$

where $V_n(\hat{\boldsymbol{\theta}}) = \frac{1}{n}\sum_{j=1}^p \boldsymbol{\epsilon}^T(\hat{\boldsymbol{\theta}}_j - \boldsymbol{\theta}_j^0)$. We wish to bound the empirical process $V_n(\hat{\boldsymbol{\theta}})$. We have

$$\begin{aligned}
\left|V_n(\hat{\boldsymbol{\theta}})\right| &= \frac{1}{\sqrt{n}}\left|\sum_{j=1}^p\left[\frac{\alpha}{\sqrt{n}}\boldsymbol{\epsilon}^T\boldsymbol{P}_j^T\boldsymbol{U}\boldsymbol{D}\boldsymbol{P}_j(\hat{\boldsymbol{\theta}}_j-\boldsymbol{\theta}_j^0) + \frac{1-\alpha}{\sqrt{n}}\boldsymbol{\epsilon}^T(\hat{\boldsymbol{\theta}}_j-\boldsymbol{\theta}_j^0)\right]\right| \\
&\le \frac{1}{\sqrt{n}}\sum_{j=1}^p\left[\left|\frac{\alpha}{\sqrt{n}}\boldsymbol{\epsilon}^T\boldsymbol{P}_j^T\boldsymbol{U}\boldsymbol{D}\boldsymbol{P}_j(\hat{\boldsymbol{\theta}}_j-\boldsymbol{\theta}_j^0)\right| + \left|\frac{1-\alpha}{\sqrt{n}}\boldsymbol{\epsilon}^T(\hat{\boldsymbol{\theta}}_j-\boldsymbol{\theta}_j^0)\right|\right] \\
&\le \frac{1}{\sqrt{n}}\sum_{j=1}^p\left[\|\boldsymbol{v}_{1j}\|_\infty\alpha\|\boldsymbol{D}\boldsymbol{P}_j(\hat{\boldsymbol{\theta}}_j-\boldsymbol{\theta}_j^0)\|_1 + \|\boldsymbol{v}_2\|_2(1-\alpha)\|\hat{\boldsymbol{\theta}}_j-\boldsymbol{\theta}_j^0\|_2\right] \\
&\le \frac{\max_{1\le j\le p}\|\boldsymbol{v}_{1j}\|_\infty}{\sqrt{n}}\sum_{j=1}^p\alpha\left(\|\boldsymbol{D}\boldsymbol{P}_j\hat{\boldsymbol{\theta}}_j\|_1 + \|\boldsymbol{D}\boldsymbol{P}_j\boldsymbol{\theta}_j^0\|_1\right) + \frac{\|\boldsymbol{v}_2\|_2}{\sqrt{n}}\sum_{j=1}^p(1-\alpha)\left(\|\hat{\boldsymbol{\theta}}_j\|_2 + \|\boldsymbol{\theta}_j^0\|_2\right)
\end{aligned}$$

where $\boldsymbol{v}_{1j} = \frac{1}{\sqrt{n}}\boldsymbol{U}^T\boldsymbol{P}_j\boldsymbol{\epsilon}$ and $\boldsymbol{v}_2 = \frac{1}{\sqrt{n}}\boldsymbol{\epsilon}$.

We now establish bounds for $\max_{1\le j\le p}\|\boldsymbol{v}_{1j}\|_\infty$ and $\|\boldsymbol{v}_2\|_2$ that hold with large probability.

Note that $\boldsymbol{v}_{1j} \in \mathbb{R}^{n-1} \sim MVN(\mathbf{0}, \frac{\sigma^2}{n}\boldsymbol{U}^T\boldsymbol{U})$. Since $\max_{1\le i\le n}(\frac{\sigma^2}{n}\boldsymbol{U}^T\boldsymbol{U})_{i,i} \le \frac{\sigma^2}{4}$, Fact A.5.1 with $c = \frac{\sigma^2}{4}$ gives $P\left(\max_{1\le j\le p}\frac{\|\boldsymbol{v}_{1j}\|_\infty}{\sqrt{n}} \ge w_1\right) \le \frac{2}{(n-1)p}$ where $w_1 = \sigma\sqrt{\frac{\log((n-1)p)}{n}}$. Now recall $\boldsymbol{v}_2 \sim MVN(\mathbf{0}, \frac{\sigma^2}{n}\boldsymbol{I})$, so by Fact A.5.2, $P\left(\frac{\|\boldsymbol{v}_2\|_2}{\sqrt{n}} \ge w_2\right) \le \frac{1}{n}$ where $w_2 = \frac{\sigma}{\sqrt{n}}\sqrt{1+\sqrt{\frac{4\log n}{n}}+\frac{4\log n}{n}}$.

Therefore, with probability at least $1 - \left(\frac{2}{(n-1)p} + \frac{1}{n}\right)$,

$$|V_n(\hat{\boldsymbol{\theta}})| \leq \sum_{j=1}^{p} [w_1 \alpha(\|\boldsymbol{D}\boldsymbol{P}_j \hat{\boldsymbol{\theta}}_j\|_1 + \|\boldsymbol{D}\boldsymbol{P}_j \boldsymbol{\theta}_j^0\|_1) + w_2(1-\alpha)(\|\hat{\boldsymbol{\theta}}_j\|_2 + \|\boldsymbol{\theta}_j^0\|_2)].$$

Let $\lambda = 2w_1$. For $p \geq 1$ and $n \geq 15$, $w_1 > w_2$. Thus, with high probability,

$$|V_n(\hat{\boldsymbol{\theta}})| \leq \frac{\lambda}{2} \sum_{j=1}^{p} [\alpha(\|\boldsymbol{D}\boldsymbol{P}_j \hat{\boldsymbol{\theta}}_j\|_1 + \|\boldsymbol{D}\boldsymbol{P}_j \boldsymbol{\theta}_j^0\|_1) + (1-\alpha)(\|\hat{\boldsymbol{\theta}}_j\|_2 + \|\boldsymbol{\theta}_j^0\|_2)] = \frac{\lambda}{2}(\Omega(\hat{\boldsymbol{\theta}}) + \Omega(\boldsymbol{\theta}^0)). \quad \text{(A.6)}$$

The result follows from plugging the bound from (A.6) into (A.5). ■

A.6 Derivations of Results from Section 2.6.2

For $\ell(\boldsymbol{\theta})$ of the form (2.17), we can calculate

$$\begin{aligned}
\dot{\ell}(\boldsymbol{\theta}) &= (\mathbf{1}_{p+1} \otimes \boldsymbol{I}_n) \left[\text{expit}((\mathbf{1}_{p+1}^T \otimes \boldsymbol{I}_n)\boldsymbol{\theta}) - \boldsymbol{y}\right] \\
\ddot{\ell}(\boldsymbol{\theta}) &= (\mathbf{1}_{p+1} \otimes \boldsymbol{I}_n) \text{diag}\left(\text{expit}((\mathbf{1}_{p+1}^T \otimes \boldsymbol{I}_n)\boldsymbol{\theta})(1 - \text{expit}((\mathbf{1}_{p+1}^T \otimes \boldsymbol{I}_n)\boldsymbol{\theta}))\right)(\mathbf{1}_{p+1}^T \otimes \boldsymbol{I}_n) \\
&\preceq \frac{1}{4}(\mathbf{1}_{p+1}\mathbf{1}_{p+1}^T \otimes \boldsymbol{I}_n) \preceq \frac{1}{4}(p+1)\boldsymbol{I}.
\end{aligned}$$

Thus, plugging in $L = \frac{1}{4}(p+1)$ and $\dot{\ell}(\boldsymbol{\theta})$ into (2.16), we now have

$$\hat{\boldsymbol{\theta}}^k = \underset{\boldsymbol{\theta} \in \mathbb{R}^{n(p+1)}}{\text{argmin}} \quad \frac{p+1}{8} \left\| \boldsymbol{\theta} - \hat{\boldsymbol{\theta}}^{k-1} + \frac{4}{p+1}(\mathbf{1}_{p+1} \otimes \boldsymbol{I}_n) \left[\text{expit}((\mathbf{1}_{p+1}^T \otimes \boldsymbol{I}_n)\hat{\boldsymbol{\theta}}^{k-1}) - \boldsymbol{y}\right] \right\|_2^2 + \lambda \sum_{j=1}^{p} Q_j(\boldsymbol{\theta}_j).$$

This is separable in $\boldsymbol{\theta}_j$, and is equivalent to (2.17) by inspection.

A.7 Proof of Lemma 2.6.1

Proof There are two main tasks:

Task 1: Derive the form of $\hat{\boldsymbol{\theta}}$, the solution to (2.19).

Task 2: Show that the solution to (2.18) is $\tilde{\boldsymbol{\theta}} = (1 - (1-\alpha)\lambda/\|\hat{\boldsymbol{\theta}}\|_2)_+ \hat{\boldsymbol{\theta}}$.

We begin with Task 1. We rewrite (2.19) as

$$\underset{\boldsymbol{\theta},\boldsymbol{z}}{\text{minimize}} \quad \frac{1}{2}\left\|\boldsymbol{y}-\boldsymbol{\theta}\right\|_2^2+\alpha\lambda\left\|\boldsymbol{z}\right\| \text{ subject to } \boldsymbol{z}=\boldsymbol{B}\boldsymbol{\theta},$$

which has Lagrangian $\mathcal{L}(\boldsymbol{\theta},\boldsymbol{z},\boldsymbol{v})=\frac{1}{2}\left\|\boldsymbol{y}-\boldsymbol{\theta}\right\|_2^2+\alpha\lambda\left\|\boldsymbol{z}\right\|+\boldsymbol{v}^T(\boldsymbol{B}\boldsymbol{\theta}-\boldsymbol{z})$. The dual function is

$$g(\boldsymbol{v})=\inf_{\boldsymbol{\theta},\boldsymbol{z}}\ \mathcal{L}(\boldsymbol{\theta},\boldsymbol{z},\boldsymbol{v})=\inf_{\boldsymbol{z}}\left\{\frac{1}{2}\left\|\boldsymbol{B}^T\boldsymbol{v}\right\|_2^2+\boldsymbol{v}^T\boldsymbol{B}(\boldsymbol{y}-\boldsymbol{B}^T\boldsymbol{v})+\alpha\lambda\left\|\boldsymbol{z}\right\|-\boldsymbol{v}^T\boldsymbol{z}\right\},$$

where the second equality follows from noting that the partial minimum with respect to $\boldsymbol{\theta}$ satisfies $\boldsymbol{\theta}=\boldsymbol{y}-\boldsymbol{B}^T\boldsymbol{v}$. Thus $g(\boldsymbol{v})=\frac{1}{2}\left\|\boldsymbol{B}^T\boldsymbol{v}\right\|_2^2+\boldsymbol{v}^T\boldsymbol{B}(\boldsymbol{y}-\boldsymbol{B}^T\boldsymbol{v})$ if $\left\|\boldsymbol{v}\right\|_*\leq\alpha\lambda$ and $-\infty$ otherwise, where $\left\|\cdot\right\|_*$ is the dual norm of $\left\|\cdot\right\|$. Finally, the dual problem is

$$\underset{\left\|\boldsymbol{v}\right\|_*\leq\alpha\lambda}{\text{maximize}} \quad -\frac{1}{2}\left\|\boldsymbol{y}-\boldsymbol{B}^T\boldsymbol{v}\right\|_2^2+\frac{1}{2}\boldsymbol{y}^T\boldsymbol{y}.$$

Letting $\hat{\boldsymbol{v}}=\underset{\left\|\boldsymbol{v}\right\|_*\leq\alpha\lambda}{\text{argmin}} \quad \left\|\boldsymbol{y}-\boldsymbol{B}^T\boldsymbol{v}\right\|_2^2$, the solution to (2.19) is $\hat{\boldsymbol{\theta}}=\boldsymbol{y}-\boldsymbol{B}^T\hat{\boldsymbol{v}}$.

We now move on to Task 2. Rewriting (2.18) as

$$\underset{\boldsymbol{\theta},\boldsymbol{z}_1,\boldsymbol{z}_2}{\text{minimize}} \quad \frac{1}{2}\left\|\boldsymbol{y}-\boldsymbol{\theta}\right\|_2^2+\alpha\lambda\left\|\boldsymbol{z}_1\right\|+(1-\alpha)\lambda\left\|\boldsymbol{z}_2\right\|_2 \text{ subject to } \boldsymbol{z}_1=\boldsymbol{B}\boldsymbol{\theta},\boldsymbol{z}_2=\boldsymbol{\theta}$$

and writing out the Lagrangian, one can show that the dual problem is

$$\underset{\left\|\boldsymbol{v}\right\|_*\leq\alpha\lambda,\left\|\boldsymbol{u}\right\|_2\leq(1-\alpha)\lambda}{\text{maximize}} \quad -\frac{1}{2}\left\|\boldsymbol{y}-\boldsymbol{B}^T\boldsymbol{v}-\boldsymbol{u}\right\|_2^2+\frac{1}{2}\boldsymbol{y}^T\boldsymbol{y}; \tag{A.7}$$

the calculations to obtain (A.7) indicate that $\boldsymbol{\theta}=\boldsymbol{y}-\boldsymbol{B}^T\boldsymbol{v}-\boldsymbol{u}$. Problem (A.7) is equivalent to $\underset{\left\|\boldsymbol{v}\right\|_*\leq\alpha\lambda,\left\|\boldsymbol{u}\right\|_2\leq(1-\alpha)\lambda}{\text{minimize}} \quad \frac{1}{2}\left\|\boldsymbol{y}-\boldsymbol{B}^T\boldsymbol{v}-\boldsymbol{u}\right\|_2^2$. Minimizing in $\boldsymbol{u}$, we have

$$\tilde{\boldsymbol{u}}=\begin{cases}\boldsymbol{y}-\boldsymbol{B}^T\boldsymbol{v} & \text{if } \left\|\boldsymbol{y}-\boldsymbol{B}^T\boldsymbol{v}\right\|_2\leq(1-\alpha)\lambda\\ (1-\alpha)\lambda\frac{\boldsymbol{y}-\boldsymbol{B}^T\boldsymbol{v}}{\left\|\boldsymbol{y}-\boldsymbol{B}^T\boldsymbol{v}\right\|_2} & \text{if } \left\|\boldsymbol{y}-\boldsymbol{B}^T\boldsymbol{v}\right\|_2>(1-\alpha)\lambda\end{cases},$$

the projection of $\boldsymbol{y}-\boldsymbol{B}^T\boldsymbol{v}$ onto the $(1-\alpha)\lambda$ ball. Thus (A.7) is equivalent to

$$\underset{\left\|\boldsymbol{v}\right\|_*\leq\alpha\lambda}{\text{minimize}} \quad \left(\left\|\boldsymbol{y}-\boldsymbol{B}^T\boldsymbol{v}\right\|_2-(1-\alpha)\lambda\right)_+,$$

which is solved by $\tilde{\boldsymbol{v}}=\underset{\left\|\boldsymbol{v}\right\|_*\leq\alpha\lambda}{\text{argmin}} \quad \left\|\boldsymbol{y}-\boldsymbol{B}^T\boldsymbol{v}\right\|_2^2$. Therefore, we have shown that $\tilde{\boldsymbol{v}}=\hat{\boldsymbol{v}}$ and $\tilde{\boldsymbol{\theta}}=\boldsymbol{y}-\boldsymbol{B}^T\tilde{\boldsymbol{v}}-\tilde{\boldsymbol{u}}$. It follows that $\tilde{\boldsymbol{\theta}}=\hat{\boldsymbol{\theta}}-\tilde{\boldsymbol{u}}=\left(1-\frac{(1-\alpha)\lambda}{\left\|\hat{\boldsymbol{\theta}}\right\|_2}\right)_+\hat{\boldsymbol{\theta}}$. ■

Appendix B

APPENDIX FOR CHAPTER 3

B.1 Notational Details

We first give an intuitive explanation of our vectorization scheme. Recall that each row of $\boldsymbol{Q} \in \mathbb{R}^{n \times q^2}$ contains $q^2 - 1$ elements that equal 0, and a single 1 that extracts an element of $\boldsymbol{m}$ according to the covariate values for that observation. For example, consider the ith row of $\boldsymbol{Q}$ for $(x_{1i}, x_{2i}) = (0.4, 0.8)$ in Figure B.1(a). These covariate values fall within the 2nd row and 3rd column of the 4×4 grid, meaning that $\boldsymbol{M}_{(2)(3)}$ provides an estimate for y_i. After vectorizing $\boldsymbol{M}$, $\boldsymbol{M}_{(2)(3)}$ is $\boldsymbol{m}_{10}$, the 10th element of the mean vector. Note that we can convert between the matrix and vector notation by taking the column number minus one multiplied by q and adding the row number (e.g., $(3-1) \times 4 + 2$). The correspondence between $\boldsymbol{M}$ and $\boldsymbol{m}$ is illustrated in Figures B.1(b) and B.1(c). Thus the ith row of $\boldsymbol{Q}$ would contains all zeros, except a single 1 for the 10th element. Finally, $(\boldsymbol{Qm})_i = m_{10}$.

Before formally defining the function Ω and matrices $\boldsymbol{Q}$, $\boldsymbol{R}_i$ for $i = 1, \ldots, q-1$, and $\boldsymbol{C}_i$ for $i = 1, \ldots, q-1$ introduced in Section 3.1.2, we define a quantile function. We use quantile$(\cdot)$ to denote the quantile range into which an element falls: quantile$(x_{1i}) = k$ if x_{1i} is between the $\frac{k-1}{q}$- and $\frac{k}{q}$-quantiles of $\boldsymbol{x}_1$. For example, if $n = q = 4$ and $\boldsymbol{x}_1 = (9\ 3\ 5\ 2)^T$, then quantile$(x_{11}) = 4$. Similarly, if $n = 6$, $q = 3$, and $\boldsymbol{x}_1 = (7\ 2\ 3\ 8\ 1\ 5)^T$, then quantile$(x_{16}) = 2$.

We define the function Ω as $\Omega(\boldsymbol{M}, x_{1i}, x_{2i}) = M_{(a)(b)}$ where $a =$ quantile(x_{1i}) and $b =$ quantile(x_{2i}).

We construct $\boldsymbol{Q} \in \mathbb{R}^{n \times q^2}$ such that

$$[\boldsymbol{Q}]_{jk} = \begin{cases} 1 & \text{if } k = \text{quantile}(x_{1j}) + q \times (\text{quantile}(x_{2j}) - 1) \\ 0 & \text{otherwise} \end{cases},$$

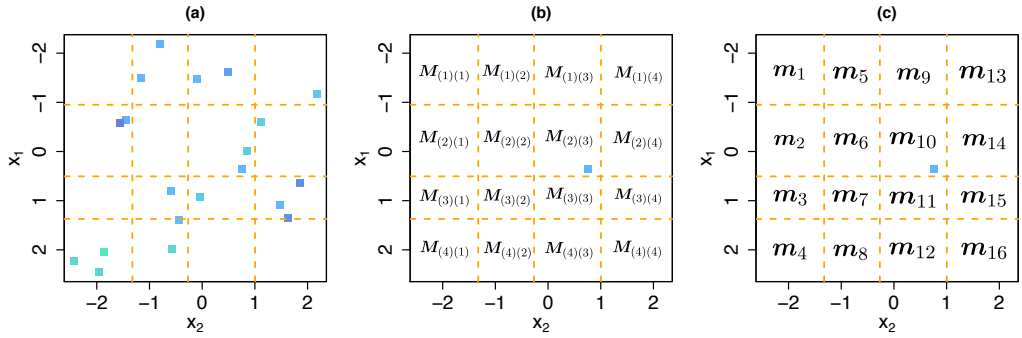


Figure B.1: In *(a)*, each of the 20 squares represents an observation (x_1, x_2, y). There are $q^2 = 16$ bins of (x_1, x_2) values, whose boundaries coincide with the quartiles (- - -) of $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$. In *(b)* and *(c)*, we label the elements of $\boldsymbol{M}$ and $\boldsymbol{m}$, respectively, corresponding to each bin of (x_1, x_2) values. Additionally, in *(b)* and *(c)*, we show $(x_{1i}, x_{2i}) = (0.4, 0.8)$, which is used in Appendix B.1 to describe the construction of $\boldsymbol{Q}$.

$\boldsymbol{R}_i \in \mathbb{R}^{q \times q^2}$ for $i = 1, \ldots, q-1$ such that

$$[\boldsymbol{R}_i]_{jk} = \begin{cases} 1 & \text{if } k = i + q \times (j-1) \\ -1 & \text{if } k = i + 1 + q \times (j-1) \\ 0 & \text{otherwise} \end{cases},$$

and $\boldsymbol{C}_i \in \mathbb{R}^{q \times q^2}$ for $i = 1, \ldots, q-1$ such that

$$[\boldsymbol{C}_i]_{jk} = \begin{cases} 1 & \text{if } k = j + q \times (i-1) \\ -1 & \text{if } k = j + q \times i \\ 0 & \text{otherwise} \end{cases}.$$

B.2 Alternative Penalties

A more general formulation of our proposal in (3.1) is

$$\underset{\boldsymbol{M} \in \mathbb{R}^{q \times q}}{\text{minimize}} \quad \frac{1}{2} \sum_{i=1}^{n} (y_i - \Omega(\boldsymbol{M}, x_{1i}, x_{2i}))^2 + \lambda \sum_{i=1}^{q-1} \left[\left\| \boldsymbol{M}_{i\cdot} - \boldsymbol{M}_{(i+1)\cdot} \right\|_t + \left\| \boldsymbol{M}_{\cdot i} - \boldsymbol{M}_{\cdot(i+1)} \right\|_t \right], \quad \text{(B.1)}$$

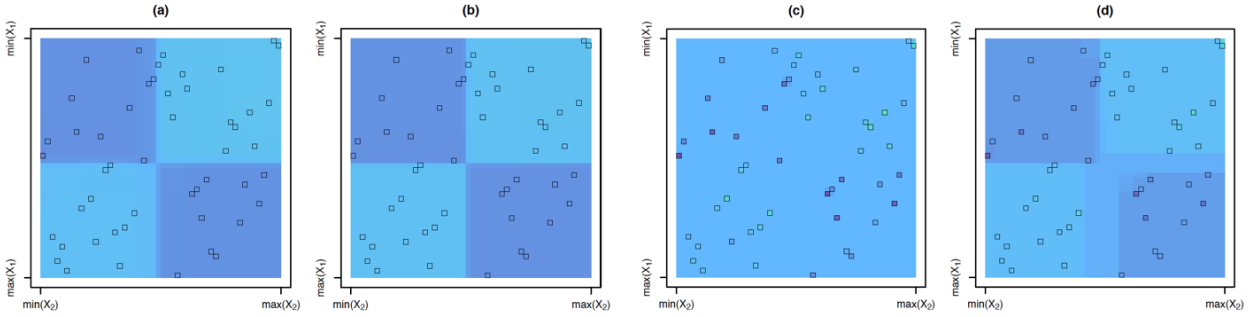


Figure B.2: The estimated mean model from solving (B.1) for *(a)* $t = 2$ (CRISP), *(b)* $t = \infty$, and *(c)* $t = 1$, as well as *(d)* the estimated mean model from solving (B.3). The methods are described in detail in Appendix B.2. Note that $q = n$ was used for all methods. Data was generated for $n = 50$ from Scenario 2 (described in Section 3.2). The locations of the 50 observations are outlined in each plot. The heat scale legend is in Figure 3.1(e).

which is equivalent to (3.1) for $t = 2$. One might consider solving (B.1) for $t = \infty$, which (like $t = 2$) encourages pairs of neighboring rows or columns of $\boldsymbol{M}$ to be identical. We compare the fit for $t = 2$ to that for $t = \infty$ in Figure B.2(a)–(b). While $t = \infty$ gives desirable fits similar to $t = 2$, the computational time required is much higher than that for $t = 2$. This is because when adapted to $t = \infty$, Step 2(b) of Algorithm 2 no longer has a closed-form solution [Duchi and Singer, 2009].

We also consider the use of $t = 1$ in (B.1); this encourages each element of $\boldsymbol{M}$ to equal its four adjacent elements. However, using $t = 1$ gives very poor results: the bins of $\boldsymbol{M}$ containing observations are estimated to be shrunken versions of their observed values, while the bins of $\boldsymbol{M}$ without observations are estimated to be a common value (Figure B.2(c)). In a sense, the penalization for $t = 1$ is too local given the data sparsity (e.g., only q of q^2 elements observed when $q = n$).

The results for $t = 1$ improve if an additional penalty is added to the objective function. First, note that (B.1) can also be written as

$$\underset{\boldsymbol{M} \in \mathbb{R}^{q \times q}}{\text{minimize}} \quad \frac{1}{2} \sum_{i=1}^{n} \left(y_i - \Omega(\boldsymbol{M}, x_{1i}, x_{2i}) \right)^2 + \lambda \left(\|\boldsymbol{M}^T \boldsymbol{D}^T\|_{t,1} + \|\boldsymbol{M} \boldsymbol{D}^T\|_{t,1} \right), \tag{B.2}$$

where $\boldsymbol{D} = [\boldsymbol{I}_{(q-1)\times(q-1)} \ \boldsymbol{0}_{(q-1)\times 1}] - [\boldsymbol{0}_{(q-1)\times 1} \ \boldsymbol{I}_{(q-1)\times(q-1)}]$. Motivated by a proposal from van de Geer [2000], we add an additional penalty to (B.2) with $t = 1$,

$$\underset{\boldsymbol{M}\in\mathbb{R}^{q\times q}}{\text{minimize}} \quad \frac{1}{2}\sum_{i=1}^{n}\left(y_i - \Omega(\boldsymbol{M}, x_{1i}, x_{2i})\right)^2 + \lambda\left(\|\boldsymbol{M}^T\boldsymbol{D}^T\|_{1,1} + \|\boldsymbol{M}\boldsymbol{D}^T\|_{1,1} + \|\boldsymbol{D}\boldsymbol{M}\boldsymbol{D}^T\|_{1,1}\right). \quad \text{(B.3)}$$

The penalty $\|\boldsymbol{D}\boldsymbol{M}\boldsymbol{D}^T\|_{1,1}$ encourages $|M_{(i)(j)} + M_{(i-1)(j-1)} - M_{(i-1)j} - M_{i(j-1)}|$ to equal zero, which results in a block structure as shown in Figure B.2(d). While (B.3) outperforms (B.1) with $t = 1$, CRISP with $t = 2$ yields better results.

B.3 Details of Algorithm 2

B.3.1 Derivation of Algorithm 2

The scaled augmented Lagrangian of (3.4) is

$$\begin{aligned} L_\rho(\boldsymbol{m}, \boldsymbol{z}, \boldsymbol{u}) = \frac{1}{2}\|\boldsymbol{y} - \boldsymbol{Q}\boldsymbol{m}\|_2^2 + \lambda\sum_{i=1}^{q-1}\left[\|\boldsymbol{z}_{1i}\|_2 + \|\boldsymbol{z}_{2i}\|_2\right] \\ + \frac{\rho}{2}\sum_{i=1}^{q-1}\left[\|\boldsymbol{R}_i\boldsymbol{m} - \boldsymbol{z}_{1i} + \boldsymbol{u}_{1i}\|_2^2 + \|\boldsymbol{C}_i\boldsymbol{m} - \boldsymbol{z}_{2i} + \boldsymbol{u}_{2i}\|_2^2\right] \end{aligned} \quad \text{(B.4)}$$

where $\boldsymbol{u} = \left((\boldsymbol{u}_{11})^T \ldots (\boldsymbol{u}_{1(q-1)})^T (\boldsymbol{u}_{21})^T \ldots (\boldsymbol{u}_{2(q-1)})^T\right)^T$ is the scaled dual variable. Solving (3.4) using ADMM relies on initializing estimates $\boldsymbol{m}^{(0)} := \boldsymbol{0}, \boldsymbol{z}^{(0)} := \boldsymbol{0}$, and $\boldsymbol{u}^{(0)} := \boldsymbol{0}$ and then iterating over three steps until convergence. At iteration k, the updates are

Step 1. $\boldsymbol{m}^{(k)} := \underset{\boldsymbol{m}}{\text{argmin}} \quad L_\rho\left(\boldsymbol{m}^{(k-1)}, \boldsymbol{z}^{(k-1)}, \boldsymbol{u}^{(k-1)}\right)$

Step 2. $\boldsymbol{z}^{(k)} := \underset{\boldsymbol{z}}{\text{argmin}} \quad L_\rho\left(\boldsymbol{m}^{(k)}, \boldsymbol{z}^{(k-1)}, \boldsymbol{u}^{(k-1)}\right)$

Step 3. $\boldsymbol{u}_{1i}^{(k)} := \boldsymbol{u}_{1i}^{(k-1)} + \boldsymbol{R}_i\boldsymbol{m}^{(k)} - \boldsymbol{z}_{1i}^{(k)}$ for $i = 1, \ldots, q-1$

$\boldsymbol{u}_{2i}^{(k)} := \boldsymbol{u}_{2i}^{(k-1)} + \boldsymbol{C}_i\boldsymbol{m}^{(k)} - \boldsymbol{z}_{2i}^{(k)}$ for $i = 1, \ldots, q-1$

Note that Step 3 can equivalently be written as $\boldsymbol{u}^{(k)} := \boldsymbol{u}^{(k-1)} + \boldsymbol{A}\boldsymbol{m}^{(k)} - \boldsymbol{z}^{(k)}$. We provide details regarding Steps 1 and 2 below.

Details of Step 1

The optimality condition of (B.4) for $\boldsymbol{m}$ is

$$\frac{\partial L_\rho}{\partial \boldsymbol{m}} = -\boldsymbol{Q}^T(\boldsymbol{y} - \boldsymbol{Q}\boldsymbol{m}) + \rho \sum_{i=1}^{q-1} \left[\boldsymbol{R}_i^T(\boldsymbol{R}_i\boldsymbol{m} - \boldsymbol{z}_{1i} + \boldsymbol{u}_{1i}) + \boldsymbol{C}_i^T(\boldsymbol{C}_i\boldsymbol{m} - \boldsymbol{z}_{2i} + \boldsymbol{u}_{2i}) \right] = \boldsymbol{0}$$

or equivalently, $-\boldsymbol{Q}^T(\boldsymbol{y} - \boldsymbol{Q}\boldsymbol{m}) + \rho\boldsymbol{A}^T(\boldsymbol{A}\boldsymbol{m} + \boldsymbol{u} - \boldsymbol{z}) = \boldsymbol{0}$. Therefore the update for Step 1 is $\boldsymbol{m}^{(k)} := \left[\boldsymbol{Q}^T\boldsymbol{Q} + \rho\boldsymbol{A}^T\boldsymbol{A}\right]^{-1}\left[\boldsymbol{Q}^T\boldsymbol{y} + \rho\boldsymbol{A}^T(\boldsymbol{z}^{(k-1)} - \boldsymbol{u}^{(k-1)})\right]$.

Details of Step 2

The *proximal operator* $\mathbf{prox}_{\lambda f}$ of λf is defined by $\mathbf{prox}_{\lambda f}(\boldsymbol{v}) = \underset{\boldsymbol{x}}{\operatorname{argmin}} \left(f(\boldsymbol{x}) + \frac{1}{2\lambda}\|\boldsymbol{x} - \boldsymbol{v}\|_2^2 \right)$. The minimization for Step 2 is separable in the $\boldsymbol{z}_{1i}$ and $\boldsymbol{z}_{2i}$ for $i = 1, \ldots, q-1$. The minimization for $\boldsymbol{z}_{1i}$ is

$$\begin{aligned}
\boldsymbol{z}_{1i}^{(k)} &:= \underset{\boldsymbol{z}_{1i}}{\operatorname{argmin}} \left[\lambda \left\|\boldsymbol{z}_{1i}\right\|_2 + \frac{\rho}{2} \left\| \boldsymbol{R}_i\boldsymbol{m}^{(k)} - \boldsymbol{z}_{1i} + \boldsymbol{u}_{1i}^{(k-1)} \right\|_2^2 \right] \\
&= \mathbf{prox}_{\frac{\lambda}{\rho}\|\cdot\|_2}\left(\boldsymbol{R}_i\boldsymbol{m}^{(k)} + \boldsymbol{u}_{1i}^{(k-1)} \right) \\
&= \left(\boldsymbol{R}_i\boldsymbol{m}^{(k)} + \boldsymbol{u}_{1i}^{(k-1)} \right) \left(1 - \frac{\lambda}{\rho \left\| \boldsymbol{R}_i\boldsymbol{m}^{(k)} + \boldsymbol{u}_{1i}^{(k-1)} \right\|_2} \right)_+ .
\end{aligned}$$

Similarly, the update for $\boldsymbol{z}_{2i}$ is $\boldsymbol{z}_{2i}^{(k)} := \left(\boldsymbol{C}_i\boldsymbol{m}^{(k)} + \boldsymbol{u}_{2i}^{(k-1)} \right) \left(1 - \frac{\lambda}{\rho \left\| \boldsymbol{C}_i\boldsymbol{m}^{(k)} + \boldsymbol{u}_{2i}^{(k-1)} \right\|_2} \right)_+$.

B.3.2 Stopping Criterion

We use the stopping criterion for Algorithm 2 suggested in Boyd et al. [2011], stopping when the primal residual $\boldsymbol{r}^{(k)} = \boldsymbol{A}\boldsymbol{m}^{(k)} - \boldsymbol{z}^{(k)}$ and dual residual $\boldsymbol{s}^{(k)} = \rho\boldsymbol{A}^T\left(\boldsymbol{z}^{(k-1)} - \boldsymbol{z}^{(k)}\right)$ are sufficiently small. Specifically, we check if

$$\|\boldsymbol{r}^{(k)}\|_2 \leq \sqrt{2q(q-1)}\epsilon^{abs} + \epsilon^{rel} \max\{\|\boldsymbol{A}\boldsymbol{m}^{(k)}\|_2, \|\boldsymbol{z}^{(k)}\|_2\} \quad \text{and} \quad \|\boldsymbol{s}^{(k)}\|_2 \leq q\epsilon^{abs} + \epsilon^{rel}\|\rho\boldsymbol{A}^T\boldsymbol{u}^{(k)}\|_2$$

with $\epsilon^{abs}, \epsilon^{rel} > 0$. We use $\epsilon^{abs} = 10^{-4}$ and $\epsilon^{rel} = 10^{-2}$ in order to obtain the results presented in Sections 3.2 and 3.5.

B.3.3 Varying Penalty Parameter

We can vary ρ from iteration to iteration in order to achieve better convergence and reduce the dependence of performance on the initially chosen ρ. We adopt the scheme for varying ρ that is reviewed in Boyd et al. [2011]. Since we use the scaled dual variable, $\boldsymbol{u}$ must also be updated in conjunction with the updating of ρ. At the end of each iteration, we apply the updates

$$(\rho^{(k+1)}, \boldsymbol{u}^{(k+1)}) := \begin{cases} (\tau^{incr}\rho^{(k)}, \boldsymbol{u}^{(k)}/\tau^{incr}) & \text{if } \|\boldsymbol{r}^{(k)}\|_2 > \delta\|\boldsymbol{s}^{(k)}\|_2 \\ (\rho^{(k)}/\tau^{decr}, \tau^{decr}\boldsymbol{u}^{(k)}) & \text{if } \|\boldsymbol{s}^{(k)}\|_2 > \delta\|\boldsymbol{r}^{(k)}\|_2 \\ (\rho^{(k)}, \boldsymbol{u}^{(k)}) & \text{otherwise} \end{cases}$$

where $\delta, \tau^{incr}, \tau^{decr} > 1$. We choose $\delta = 10$ and $\tau^{incr} = \tau^{decr} = 2$. Updating ρ keeps the norms of the residuals $\boldsymbol{r}^{(k)}$ and $\boldsymbol{s}^{(k)}$ within a factor of δ of one another. While convergence of ADMM has only been proven for fixed ρ, varying ρ has been shown to work well in practice [Boyd et al., 2011].

B.3.4 Modification to Provide Sparsity

Inspection of the updates for $\boldsymbol{z}_{1i}^*$ and $\boldsymbol{z}_{2i}^*$ in Algorithm 2 indicates that the ADMM algorithm yields sparsity in $\boldsymbol{z}_{1i}^*$ and $\boldsymbol{z}_{2i}^*$, but not necessarily exact equality of the rows and columns of $\boldsymbol{M}^*$. This is in effect a numerical issue: our algorithm might yield $\boldsymbol{z}_{1i} = \boldsymbol{0}$, but $\|\boldsymbol{M}_{i\cdot}^* - \boldsymbol{M}_{(i+1)\cdot}^*\|_2 = 1 \times 10^{-8}$. To resolve this issue, we first determine the "blocks" of $\boldsymbol{m}^*$ using an initial run of Algorithm 2, and then solve (3.4) once more with constraints on the rows and columns of $\boldsymbol{M}$ to enforce equality of the appropriate rows and columns. This second optimization is performed simply to yield an estimate of $\boldsymbol{M}$ for which elements are exactly equal within each block.

B.4 Details of Simulations in Section 3.2

The mean models $f(x_1, x_2)$ used to generate data for Scenarios 1–4 in Section 3.2 are defined as follows. Note that x_1 and x_2 are sampled uniformly from $[-2.5, 2.5]$. We define the indicator function $\mathbf{1}_{\mathcal{A}}(x) = \begin{cases} 1 & \text{if } x \in \mathcal{A} \\ 0 & \text{otherwise} \end{cases}$.

Scenario 1: $f(x_1, x_2) = \text{sign}(x_1) \times \mathbf{1}_{[0,\infty)}(x_1 \times x_2)$

Scenario 2: $f(x_1, x_2) = -\text{sign}(x_1 \times x_2)$

Scenario 3: $f(x_1, x_2) = -3 \times \mathbf{1}_{[-2.5,-0.83)}(x_1) \times \mathbf{1}_{[-2.5,-1.25)}(x_2) + \mathbf{1}_{[-2.5,-0.83)}(x_1) \times \mathbf{1}_{[-1.25,2.5]}(x_2) - 2 \times \mathbf{1}_{[-0.83,0.83]}(x_1) \times \mathbf{1}_{[-2.5,0)}(x_2) + 2 \times \mathbf{1}_{[-0.83,0.83]}(x_1) \times \mathbf{1}_{[0,2.5]}(x_2) - \mathbf{1}_{(0.83,2.5]}(x_1) \times \mathbf{1}_{[-2.5,1.25)}(x_2) + 3 \times \mathbf{1}_{(0.83,2.5]}(x_1) \times \mathbf{1}_{[1.25,2.5]}(x_2)$

Scenario 4: $f(x_1, x_2) = \frac{10}{\left(\frac{x_1-2.5}{3}\right)^2+\left(\frac{x_2-2.5}{3}\right)^2+1} + \frac{10}{\left(\frac{x_1+2.5}{3}\right)^2+\left(\frac{x_2+2.5}{3}\right)^2+1}$

Each of the mean models $f(x_1, x_2)$ defined above is centered and scaled such that $\int_{-2.5}^{2.5}\int_{-2.5}^{2.5} f(x_1, x_2)\, dx_1\, dx_2 = 0$ and $\frac{1}{25}\int_{-2.5}^{2.5}\int_{-2.5}^{2.5} f(x_1, x_2)^2\, dx_1\, dx_2 = 2$.

B.5 Proof Sketch of Proposition 3.3.1

Proof Using the dual problem of (3.5) and Lemma 1 of Tibshirani and Taylor [2012], it can be shown that $g : \mathbb{R}^n \to \mathbb{R}^n$ with $\hat{\boldsymbol{y}} = g(\boldsymbol{y}) = (g_1(\boldsymbol{y}), \ldots, g_n(\boldsymbol{y}))^T$ is continuous and almost differentiable. Thus, Stein's lemma implies that $\text{df}(\hat{\boldsymbol{y}}) = \text{E}\left[\text{Tr}\left(\frac{\partial g(\boldsymbol{y})}{\partial \boldsymbol{y}}\right)\right]$. At the optimum of (3.5), we have

$$\boldsymbol{Q}^T(\boldsymbol{y} - \boldsymbol{Q}\boldsymbol{m}^*) = \lambda \sum_{i=1}^{q-1} [\boldsymbol{R}_i^T S_1(\boldsymbol{R}_i, \boldsymbol{m}^*) + \boldsymbol{C}_i^T S_1(\boldsymbol{C}_i, \boldsymbol{m}^*)] + \gamma \boldsymbol{m}^*, \tag{B.5}$$

where $S_1(\boldsymbol{A}_i, \boldsymbol{m}^*) = \begin{cases} \frac{\boldsymbol{A}_i \boldsymbol{m}^*}{\|\boldsymbol{A}_i \boldsymbol{m}^*\|_2} & \text{if } \|\boldsymbol{A}_i \boldsymbol{m}^*\|_2 \neq 0 \\ \in \{\boldsymbol{g} : \|\boldsymbol{g}\|_2 \leq 1\} & \text{if } \|\boldsymbol{A}_i \boldsymbol{m}^*\|_2 = 0 \end{cases}$.

We define $\mathcal{C} = \{\boldsymbol{A}_i : \|\boldsymbol{A}_i \boldsymbol{m}^*\|_2 = 0\}$ where $\boldsymbol{A}_1 = \boldsymbol{R}_1, \boldsymbol{A}_2 = \boldsymbol{R}_2, \ldots, \boldsymbol{A}_{q-1} = \boldsymbol{R}_{q-1}, \boldsymbol{A}_q = \boldsymbol{C}_1, \boldsymbol{A}_{q+1} = \boldsymbol{C}_2, \ldots, \boldsymbol{A}_{2q-2} = \boldsymbol{C}_{q-1}$. We define $\boldsymbol{A}_*$ to be the submatrix of $\boldsymbol{A}$ with the rows

corresponding to $\boldsymbol{A}_i \notin \mathcal{C}$ removed, and let $\boldsymbol{P} = \boldsymbol{I}_{q^2} - \boldsymbol{A}_*^+ \boldsymbol{A}_*$, the projection onto the space orthogonal to the row space of $\boldsymbol{A}_*$. We left-multiply (B.5) by $\boldsymbol{P}$ to give

$$\boldsymbol{P}\boldsymbol{Q}^T(\boldsymbol{y} - \boldsymbol{Q}\boldsymbol{m}^*) = \lambda \boldsymbol{P} \sum_{i:\boldsymbol{A}_i \notin \mathcal{C}} \frac{\boldsymbol{A}_i^T \boldsymbol{A}_i \boldsymbol{m}^*}{\|\boldsymbol{A}_i \boldsymbol{m}^*\|_2} + \gamma \boldsymbol{P}\boldsymbol{m}^*, \tag{B.6}$$

since $\boldsymbol{P}\boldsymbol{A}_i^T S_1(\boldsymbol{A}_i, \boldsymbol{m}^*) = \boldsymbol{0}$ if $\boldsymbol{A}_i \in \mathcal{C}$ (i.e., $\|\boldsymbol{A}_i \boldsymbol{m}^*\|_2 = 0$). Because $\boldsymbol{P}\boldsymbol{m}^* = \boldsymbol{m}^*$, (B.6) can be rewritten as

$$\boldsymbol{P}\boldsymbol{Q}^T(\boldsymbol{y} - \boldsymbol{Q}\boldsymbol{P}\boldsymbol{m}^*) = \lambda \boldsymbol{P} \sum_{i:\boldsymbol{A}_i \notin \mathcal{C}} \frac{\boldsymbol{A}_i^T \boldsymbol{A}_i \boldsymbol{P}\boldsymbol{m}^*}{\|\boldsymbol{A}_i \boldsymbol{m}^*\|_2} + \gamma \boldsymbol{m}^*. \tag{B.7}$$

We let $\boldsymbol{D} = \text{diag}\left(h(m_1^*), \cdots, h(m_{q^2}^*)\right)$, where $h(m_i^*)$ is defined to be the ratio of the number of observations in the block of $\boldsymbol{M}^*$ that contains m_i^* to the number of elements of $\boldsymbol{M}^*$ in the block of $\boldsymbol{M}^*$ that contains m_i^*. Note that $\boldsymbol{P}\boldsymbol{Q}^T\boldsymbol{Q}\boldsymbol{P} = \boldsymbol{D}\boldsymbol{P}$. Thus $\boldsymbol{P}\boldsymbol{Q}^T\boldsymbol{Q}\boldsymbol{P}\boldsymbol{m}^* = \boldsymbol{D}\boldsymbol{m}^*$, and (B.7) is equivalent to

$$\boldsymbol{P}\boldsymbol{Q}^T\boldsymbol{y} = \boldsymbol{D}\boldsymbol{m}^* + \lambda \boldsymbol{P} \sum_{i:\boldsymbol{A}_i \notin \mathcal{C}} \frac{\boldsymbol{A}_i^T \boldsymbol{A}_i \boldsymbol{P}\boldsymbol{m}^*}{\|\boldsymbol{A}_i \boldsymbol{m}^*\|_2} + \gamma \boldsymbol{m}^*. \tag{B.8}$$

We conjecture that there is a neighborhood around almost every $\boldsymbol{y}$ such that the blocks of $\boldsymbol{m}^*$ do not change. That is, $\mathcal{C}$ and $\boldsymbol{P}$ in (B.8) are constant with respect to $\boldsymbol{y}$, and the derivative of (B.8) with respect to $\boldsymbol{y}$ is

$$\boldsymbol{P}\boldsymbol{Q}^T = \left(\boldsymbol{D} + \lambda \boldsymbol{P} \sum_{i:\boldsymbol{A}_i \notin \mathcal{C}} S_2(\boldsymbol{A}_i, \boldsymbol{m}^*) \boldsymbol{P} + \gamma \boldsymbol{I} \right) \frac{\partial \boldsymbol{m}^*}{\partial \boldsymbol{y}}, \tag{B.9}$$

where $S_2(\boldsymbol{A}_i, \boldsymbol{m}^*) = \frac{\boldsymbol{A}_i^T \boldsymbol{A}_i}{\|\boldsymbol{A}_i \boldsymbol{m}^*\|_2} - \frac{\boldsymbol{A}_i^T \boldsymbol{A}_i \boldsymbol{m}^* \boldsymbol{m}^{*T} \boldsymbol{A}_i^T \boldsymbol{A}_i}{\|\boldsymbol{A}_i \boldsymbol{m}^*\|_2^3}$. Recall $\hat{\boldsymbol{y}} = \boldsymbol{Q}\boldsymbol{m}^*$, so solving (B.9) for $\frac{\partial \boldsymbol{m}^*}{\partial \boldsymbol{y}}$ and left-multiplying by $\boldsymbol{Q}$ gives

$$\frac{\partial \hat{\boldsymbol{y}}}{\partial \boldsymbol{y}} = \boldsymbol{Q} \left(\boldsymbol{D} + \lambda \boldsymbol{P} \sum_{i:\boldsymbol{A}_i \notin \mathcal{C}} S_2(\boldsymbol{A}_i, \boldsymbol{m}^*) \boldsymbol{P} + \gamma \boldsymbol{I} \right)^{-1} \boldsymbol{P}\boldsymbol{Q}^T,$$

where $\left(\boldsymbol{D} + \lambda \boldsymbol{P} \sum_{i:\boldsymbol{A}_i \notin \mathcal{C}} S_2(\boldsymbol{A}_i, \boldsymbol{m}^*) \boldsymbol{P} + \gamma \boldsymbol{I}\right)$ is invertible as both $\boldsymbol{D}$ and $\lambda \boldsymbol{P} \sum_{i:\boldsymbol{A}_i \notin \mathcal{C}} S_2(\boldsymbol{A}_i, \boldsymbol{m}^*) \boldsymbol{P}$ are positive semi-definite. Therefore, the degrees of freedom

is

$$\mathrm{E}\left[\mathrm{Tr}\left(\boldsymbol{Q}\left(\boldsymbol{D}+\lambda\boldsymbol{P}\sum_{i:\boldsymbol{A}_i\notin\mathcal{C}}S_2(\boldsymbol{A}_i,\boldsymbol{m}^*)\boldsymbol{P}+\gamma\boldsymbol{I}\right)^{-1}\boldsymbol{P}\boldsymbol{Q}^T\right)\right].$$

This establishes the unbiasedness of the estimator (3.6). ■

B.6 Proof of Corollary 3.3.2

Proof This corollary pertains to the setting in which either all rows of $\boldsymbol{M}^*$ are equal (i.e., $\boldsymbol{R}_i \in \mathcal{C}$ for all i) or all columns of $\boldsymbol{M}^*$ are equal (i.e., $\boldsymbol{C}_i \in \mathcal{C}$ for all i). In this setting, we will show $\boldsymbol{P}S_2(\boldsymbol{A}_i, \boldsymbol{m}^*) = \boldsymbol{0}$ for any $\boldsymbol{A}_i \notin \mathcal{C}$ using two facts: (1) $\boldsymbol{A}_i\boldsymbol{m}^* = c_i\boldsymbol{1}_q$ for some $c_i \in \mathbb{R}$ and (2) $\boldsymbol{P}\boldsymbol{A}_i^T = \boldsymbol{v}_i\boldsymbol{1}_q^T$ for some $\boldsymbol{v}_i \in \mathbb{R}^{q^2}$. These facts follow from the assumption that either all rows or all columns of $\boldsymbol{M}^*$ are equal. Consider some $\boldsymbol{A}_i \notin \mathcal{C}$. We have

$$\begin{aligned}
\boldsymbol{P}S_2(\boldsymbol{A}_i,\boldsymbol{m}^*) &= \frac{\boldsymbol{P}\boldsymbol{A}_i^T\boldsymbol{A}_i}{\|\boldsymbol{A}_i\boldsymbol{m}^*\|_2} - \frac{\boldsymbol{P}\boldsymbol{A}_i^T\boldsymbol{A}_i\boldsymbol{m}^*\boldsymbol{m}^{*T}\boldsymbol{A}_i^T\boldsymbol{A}_i}{\|\boldsymbol{A}_i\boldsymbol{m}^*\|_2^3} \\
&= \frac{\boldsymbol{P}\boldsymbol{A}_i^T\boldsymbol{A}_i}{\|\boldsymbol{A}_i\boldsymbol{m}^*\|_2} - \frac{(\boldsymbol{v}_i\boldsymbol{1}_q^T)(c_i\boldsymbol{1}_q)(c_i\boldsymbol{1}_q^T)\boldsymbol{A}_i}{c_i^2 q\|\boldsymbol{A}_i\boldsymbol{m}^*\|_2} \\
&= \frac{\boldsymbol{P}\boldsymbol{A}_i^T\boldsymbol{A}_i}{\|\boldsymbol{A}_i\boldsymbol{m}^*\|_2} - \frac{\boldsymbol{v}_i\boldsymbol{1}_q^T\boldsymbol{A}_i}{\|\boldsymbol{A}_i\boldsymbol{m}^*\|_2} \\
&= \frac{\boldsymbol{P}\boldsymbol{A}_i^T\boldsymbol{A}_i}{\|\boldsymbol{A}_i\boldsymbol{m}^*\|_2} - \frac{\boldsymbol{P}\boldsymbol{A}_i^T\boldsymbol{A}_i}{\|\boldsymbol{A}_i\boldsymbol{m}^*\|_2} \\
&= \boldsymbol{0}.
\end{aligned}$$

Therefore, the estimator (3.6) with $\gamma = 0$ simplifies to $\mathrm{Tr}[\boldsymbol{Q}\boldsymbol{D}^{-1}\boldsymbol{P}\boldsymbol{Q}^T] = \mathrm{Tr}[\boldsymbol{D}^{-1}\boldsymbol{P}\boldsymbol{Q}^T\boldsymbol{Q}]$. Recall that $\boldsymbol{D}$ is a diagonal matrix with $D_{ii} = h(m_i^*) = N_i^0/N_i$, where N_i^0 and N_i are the number of observations and the number of elements, respectively, in the block of $\boldsymbol{M}^*$ containing m_i^*. Note that $(\boldsymbol{P}\boldsymbol{Q}^T\boldsymbol{Q})_{ii}$ equals n_i^0/N_i, where n_i^0 is the number of observations corresponding to m_i^*. Thus

$$\mathrm{Tr}[\boldsymbol{D}^{-1}\boldsymbol{P}\boldsymbol{Q}^T\boldsymbol{Q}] = \sum_{i=1}^{q^2}\frac{(\boldsymbol{P}\boldsymbol{Q}^T\boldsymbol{Q})_{ii}}{D_{ii}} = \sum_{i:m_i^* \text{ observed}}\frac{N_i}{N_i^0}\frac{n_i^0}{N_i} = \sum_{i:m_i^* \text{ observed}}\frac{n_i^0}{N_i^0},$$

which equals the total number of blocks of $\boldsymbol{M}^*$ since the n_i^0's for a block sum to N_i^0. ■

B.7 Proof of Lemma 3.3.3

Proof If $\boldsymbol{m}^* = \left(\frac{1}{n}\mathbf{1}_n^T\boldsymbol{y}\right)\mathbf{1}_{q^2}$ solves (3.3), then there exist q-vectors $\boldsymbol{d}_{1i}, \boldsymbol{d}_{2i}$ with $\|\boldsymbol{d}_{1i}\|_2 \leq \lambda$ and $\|\boldsymbol{d}_{2i}\|_2 \leq \lambda$ such that

$$\boldsymbol{Q}^T\left(\boldsymbol{y} - \left(\frac{1}{n}\mathbf{1}_n^T\boldsymbol{y}\right)\mathbf{1}_q\right) = \sum_{i=1}^{q-1}\left[\boldsymbol{R}_i^T\boldsymbol{d}_{1i} + \boldsymbol{C}_i^T\boldsymbol{d}_{2i}\right], \tag{B.10}$$

since $\boldsymbol{Q}\mathbf{1}_{q^2} = \mathbf{1}_q$. Let $\boldsymbol{d} = (\boldsymbol{d}_{11}^T \ \cdots \ \boldsymbol{d}_{1(q-1)}^T \ \boldsymbol{d}_{21}^T \ \ldots \ \boldsymbol{d}_{2(q-1)}^T)^T$. Then (B.10) can be rewritten as

$$\boldsymbol{Q}^T\left(\boldsymbol{y} - \left(\frac{1}{n}\mathbf{1}_n^T\boldsymbol{y}\right)\mathbf{1}_q\right) = \boldsymbol{A}^T\boldsymbol{d}. \tag{B.11}$$

Note that $\boldsymbol{m}^* = \left(\frac{1}{n}\mathbf{1}_n^T\boldsymbol{y}\right)\mathbf{1}_{q^2}$ for a certain λ if and only if (B.11) is satisfied for some $\boldsymbol{d}$ for which $\|\boldsymbol{d}_{1i}\|_2 \leq \lambda, \|\boldsymbol{d}_{2i}\|_2 \leq \lambda$ for $i = 1, \ldots, q-1$. We find the $\boldsymbol{d}^*$ corresponding to the minimum λ for which $\boldsymbol{m}^* = \left(\frac{1}{n}\mathbf{1}_n^T\boldsymbol{y}\right)\mathbf{1}_{q^2}$ by solving the convex optimization problem

$$\underset{\boldsymbol{d}}{\text{minimize}} \quad \max_{1\leq i\leq q-1}\left\{\|\boldsymbol{d}_{1i}\|_2, \|\boldsymbol{d}_{2i}\|_2\right\} \quad \text{subject to} \quad \boldsymbol{Q}^T\left(\boldsymbol{y} - \left(\frac{1}{n}\mathbf{1}_n^T\boldsymbol{y}\right)\mathbf{1}_q\right) = \boldsymbol{A}^T\boldsymbol{d}.$$

Thus $\boldsymbol{m}^* = \left(\frac{1}{n}\mathbf{1}_n^T\boldsymbol{y}\right)\mathbf{1}_{q^2}$ if and only if $\lambda \geq \max_{1\leq i\leq q-1}\left\{\|\boldsymbol{d}_{1i}^*\|_2, \|\boldsymbol{d}_{2i}^*\|_2\right\}$. ■

B.8 Simulations Illustrating Performance of (3.3) *with* $\lambda = 0$ *and Variable* q

We illustrate how (3.3) with $\lambda = 0$ over a range of q values performs compared to CRISP for a variety of scenarios. We generate data with $n = 100$ by independently sampling each element of $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ from a Unif$[-2.5, 2.5]$ distribution, and then taking $\boldsymbol{y} = f(\boldsymbol{x}_1, \boldsymbol{x}_2) + \boldsymbol{\epsilon}$, where $\boldsymbol{\epsilon} \sim \text{MVN}(\mathbf{0}, \boldsymbol{I}_n)$. The four mean models $f(\boldsymbol{x}_1, \boldsymbol{x}_2)$ we consider are shown in Figure B.3. Note that these are the same mean models we consider extensively in Section 3.2.

For each mean model, we generate 1000 replicates of data and estimate the mean model using (3.3) with $\lambda = 0$ and various q. We plot the MSE, squared bias, and variance of the

mean model estimate as a function of q in Figure B.3. In Scenarios 1 and 2, $q = 2$ has the best performance, which is unsurprising given the mean model structure. Using $q = 2$, there will be four bins whose boundaries roughly coincide with the true boundaries of the mean model. As q increases, the bias increases in an oscillating fashion where even values of q give better performance than odd ones. This is because odd values of q will not tend to have bins with boundaries that coincide with the true boundaries the mean model. As q increases, most of the q^2 bins will not have observations in them, and their estimates will be the mean of $\boldsymbol{y}$. Thus the variance decreases as many bins take on the same value, but the squared bias continues to increase. In Scenarios 3 and 4, the minimum MSE occurs at $q = 4$, not $q = 2$ as in Scenarios 1 and 2. This is because the mean models in Scenarios 3 and 4 are more complex and not well-estimated using only 2×2 grid of bins.

We also consider the performance for an additional mean model, shown in Figure B.4(a). The same simulation set-up was used as for Scenarios 1–4 above. Though the blocks of the true mean model perfectly align with a grid that has $q = 3$, there are only 4 distinct blocks. The method of (3.3) with $\lambda = 0$ unsurprisingly has the best performance for $q = 3$, which is shown in Figure B.4(d). The estimated mean model from using $q = 3$ and $\lambda = 0$ has $q^2 = 9$ blocks, as shown in Figure B.4(b), since there is no adaptive shrinking together of blocks. However, CRISP is able to adaptively determine that only 4 blocks are needed, as shown in the estimated mean model in Figure B.4(c). This example illustrates how CRISP is able to adaptively determine the amount of granularity over the covariate space. With $\lambda = 0$, the amount of granularity is constant across the covariate space.

B.9 Details of Data Application

In Section 3.5, we analyze housing data with the outcome of median house value and predictors of median income and average occupancy. We plot median income versus average occupancy in Figure B.5. Note that 37 neighborhoods had an average occupancy larger than 10 and are omitted from the plot. The mean of average occupancy for these neighborhoods with an average occupancy greater than 10 was 88. In Figure B.5, we outline the central 95%

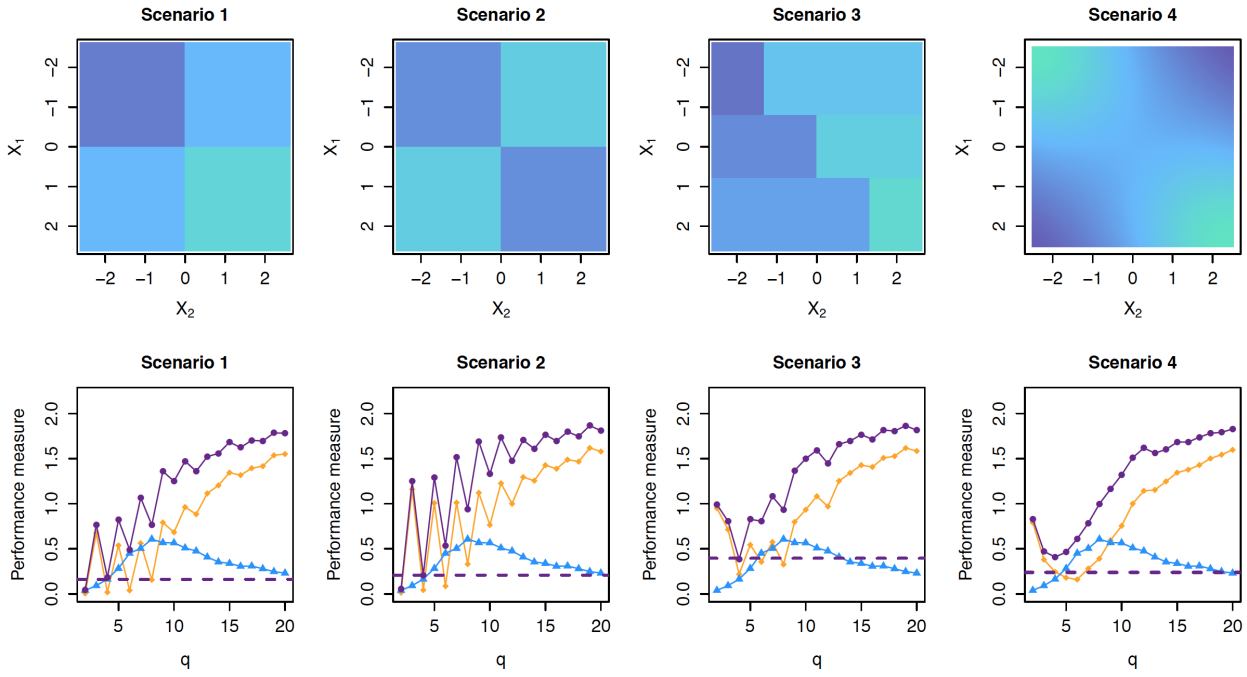


Figure B.3: The top row of figures shows the mean models $f(x_1, x_2)$ used to generate data in each of the four scenarios in Appendix B.8. The bottom row of figures shows the performance of the method of (3.3) with $\lambda = 0$ as a function of q in terms of MSE (—●—), squared bias (—◆—), and variance (—▲—). The MSE for CRISP with $q = n$ and optimal λ is shown (- - -) for comparison.

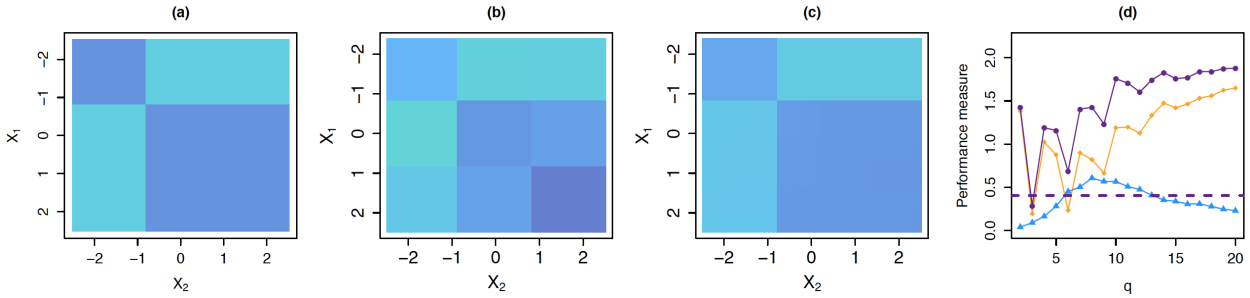


Figure B.4: In *(a)*, we plot the mean model $f(x_1, x_2)$ used to generate data for the simulation described in Appendix B.8. In *(b)*, we show the estimated mean model from the method of (3.3) with $\lambda = 0$ and $q = 3$. In *(c)*, we show the estimated mean model from CRISP with $q = n$. In *(d)*, we show the performance of the method of (3.3) with $\lambda = 0$ as a function of q in terms of MSE (—●—), squared bias (—◆—), and variance (—▲—). The MSE for CRISP with $q = n$ and optimal λ is shown (- - -) for comparison.

of the data in both covariates. That is, the 2.5% and 97.5% quantiles are shown for both covariates. We restrict our analysis to observations that fall in the central 95% of the data for both covariates. Of the original 20,640 neighborhoods, this excludes 1978 observations, leaving 18,662 observations for analysis.

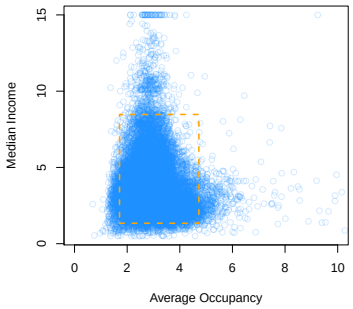


Figure B.5: We plot median income versus average occupancy for the housing data considered in Section 3.5 and described in Appendix B.9. The rectangle (- - -) identifies observations falling within the central 95% of the data for both covariates.

Appendix C

APPENDIX FOR CHAPTER 4

C.1 Data Pre-Processing

Prior to using SCALPEL to analyze the data, we perform three pre-processing steps on the raw data. These are briefly described in Step 0 of Section 4.3. First, we smooth the raw $P \times T$ data matrix spatially and temporally using a Gaussian kernel smoother with a bandwidth of 1 pixel [Hastie et al., 2009]. Second, we adjust for any bleaching effect over time. Specifically, we fit a smoothing spline with 10 degrees of freedom to the median fluorescence for each frame over time, and subtract the frame-specific smoothed median from the corresponding frame. An example of a smoothing spline fit for one calcium imaging video is shown in Figure C.1.

Finally, we apply a slight variation of the often-used $\Delta f/f$ transformation [Ahrens et al., 2013, Grewe et al., 2010, Grienberger and Konnerth, 2012]. For the ith pixel in the jth frame, the standardized fluorescence is equal to

$$y_{i,j} \equiv \frac{y^0_{i,j} - \text{median}_{t=1,\ldots,T}(y^0_{i,t})}{\text{median}_{t=1,\ldots,T}(y^0_{i,t}) + \text{quantile}_{10\%}(\boldsymbol{Y}^0)},$$

where $y^0_{i,j}$ is the fluorescence (after smoothing and bleaching correction) of the ith pixel in the jth frame. This differs from the typical $\Delta f/f$ transformation in that (1) we standardize using the median across image frames instead of the mean; and (2) we add a small number to the denominator. This adjustment in the denominator prevents small fluctuations in the amount of fluorescence at pixels with very little overall fluorescence from resulting in extremely high standardized fluorescences. In Figure C.2, we show the resulting images after each stage of pre-processing for a sample frame.

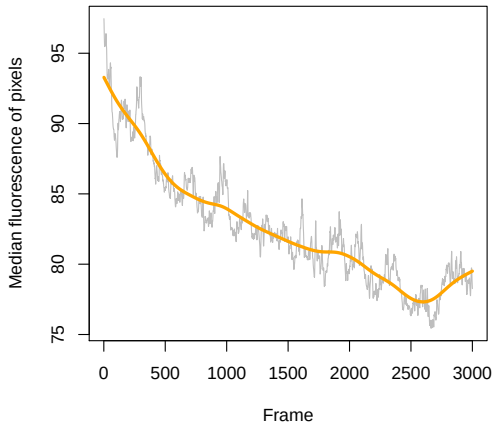


Figure C.1: We plot the median fluorescence of the pixels within each frame, along with the smoothing spline fit (▬) that is used to correct for the bleaching effect.

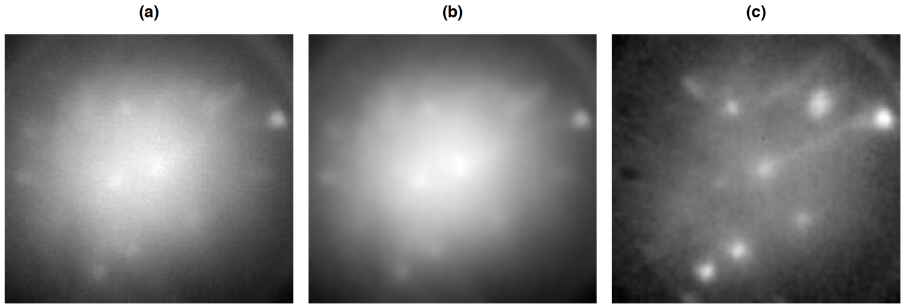


Figure C.2: In (a), we show a sample frame from the raw calcium imaging video. In (b), we show the same frame after spatial and temporal smoothing has been done and the bleaching effect has been removed. In (c), we show the frame after the $\Delta f/f$ transformation has been performed, which is the final result of the pre-processing.

C.2 Alternative Spatial Dissimilarity Metrics

We consider three alternatives to the spatial dissimilarity proposed in (4.3):

$$d_{i,j}^{union} = 1 - \frac{p_{i,j}}{p_{i,i} + p_{j,j} - p_{i,j}},\ d_{i,j}^{min} = 1 - \frac{p_{i,j}}{\min(p_{i,i}, p_{j,j})},\ \text{and}\ d_{i,j}^{max} = 1 - \frac{p_{i,j}}{\max(p_{i,i}, p_{j,j})}. \quad \text{(C.1)}$$

We refer to the dissimilarities in (C.1) as the union, min, and max dissimilarities, respectively. Note that all of the dissimilarities in (4.3) and (C.1) take on values in $[0, 1]$.

To see why we prefer the dissimilarity measure in (4.3) over those in (C.1), consider Figure C.3(d), in which one component is located entirely within the other, and the two are of substantially different sizes. Min dissimilarity assigns a value of 0 in this case, despite the fact that one component is much smaller than the other. Union dissimilarity and max dissimilarity yield a very large value of 0.62, despite the fact that there is substantial overlap between the two components. In contrast, the dissimilarity in (4.3) yields a modest but non-zero dissimilarity, which seems reasonable given that the two components in Figure C.3(d) are similar but not identical. More generally, the dissimilarity in (4.3) tends to yield reasonable values across a range of settings (Figures C.3(a)-(c)).

Often Euclidean distance, $\|\boldsymbol{a}_{\cdot,i}^0 - \boldsymbol{a}_{\cdot,j}^0\|_2$, is used as the distance metric in clustering. Here, the squared Euclidean distance equals the number of differing pixels between components. This is problematic as a distance metric for our purposes — it treats $a_{n,i}^0 = a_{n,j}^0 = 0$ the same as $a_{n,i}^0 = a_{n,j}^0 = 1$, though the latter is much more informative in terms of the components' similarity. For example, we see that the pairs of components in Figures C.3(b) and (c) have similar Euclidean distances, though the components in Figure C.3(b) are non-overlapping and could be arbitrarily far apart without affecting the Euclidean distance.

C.3 Example of a Cluster in Step 2

To see how the representative spatial component relates to the other candidate spatial components in the cluster, we give an example in Figure C.4.

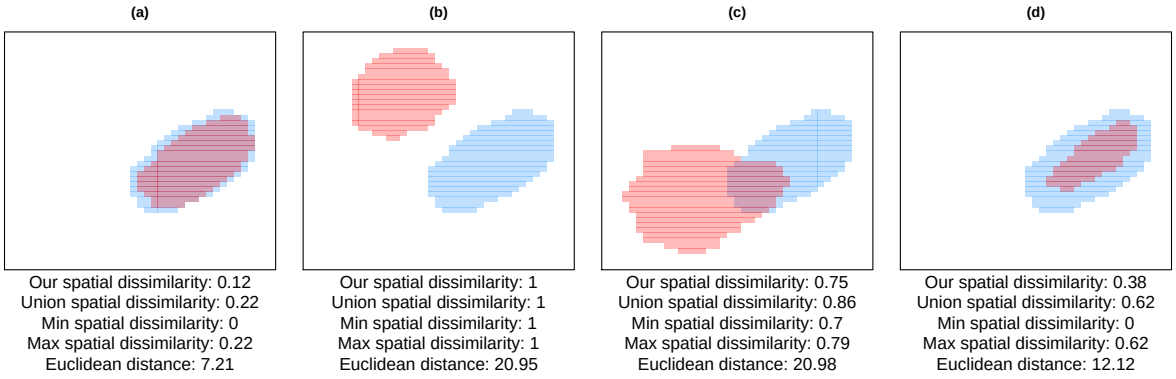


Figure C.3: We report our spatial dissimilarity metric, as defined in (4.3), and three alternative metrics, defined in (C.1), for four pairs of candidate spatial components. We also report the Euclidean distance between each pair of components.

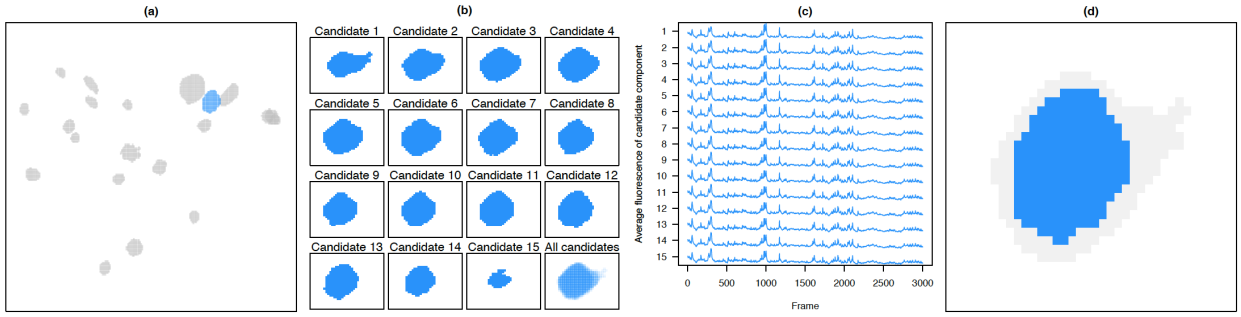


Figure C.4: We focus on a single cluster of candidate spatial components. The representative spatial component for this cluster is highlighted in (a). The spatial maps for the 15 candidate spatial components are shown in (b), along with a spatial map containing all 15 components in which the hue intensity at a given pixel indicates the number of candidate spatial components containing that pixel. In (c), we plot the corresponding average fluorescence for each of the candidate spatial components from (b). Finally, in (d), we show the representative spatial component, which is candidate spatial component 12. The gray coloring indicates the union of all candidate spatial components.

C.4 Selecting λ for (4.6)

To choose λ for (4.6) via a training/validation set approach, we perform the following steps:

1. Obtain $\tilde{\boldsymbol{A}} \in \mathbb{R}^{P \times K}$ by dividing the kth column of $\boldsymbol{A}$ by $\|\boldsymbol{a}_{\cdot,k}\|_2^2$.

2. Construct a training set $\mathcal{T}$ by sampling 60% of the pixels in each overlapping group of neurons. That is, we sample 60% of the elements in $\mathcal{M}(\mathcal{N}_1), \mathcal{M}(\mathcal{N}_2), \ldots, \mathcal{M}(\mathcal{N}_S)$. Assign the remaining pixels to the validation set, $\mathcal{V} = \{v \in (1, \ldots, P) : v \notin \mathcal{T}\}$.

3. Using Algorithm 4, solve (4.6) on the training set of pixels for a decreasing sequence of 20 λ values, $\lambda_1, \ldots, \lambda_{20}$:

$$\hat{\boldsymbol{Z}}(\lambda_i) = \underset{\boldsymbol{Z} \in \mathbb{R}^{K \times T}, \boldsymbol{Z} \geq 0}{\operatorname{argmin}} \quad \frac{1}{2} \left\| \boldsymbol{Y}_{\mathcal{T},\cdot} - \tilde{\boldsymbol{A}}_{\mathcal{T},\cdot} \boldsymbol{Z} \right\|_F^2 + \lambda_i \alpha \mathbf{1}^T \boldsymbol{Z} \mathbf{1} + \lambda_i (1 - \alpha) \sum_{k=1}^{K} \|\boldsymbol{z}_{k,\cdot}\|_2 .$$

4. For each λ_i, calculate the validation error,

$$err_{\mathcal{V}}(\lambda_i) = \frac{1}{|\mathcal{V}|} \left\| \boldsymbol{Y}_{\mathcal{V},\cdot} - \tilde{\boldsymbol{A}}_{\mathcal{V},\cdot} \hat{\boldsymbol{Z}}(\lambda_i) \right\|_F^2 .$$

   Select the optimal value of λ as

$$\lambda^* = \underset{\lambda_i}{\operatorname{argmax}} \left\{ \lambda_i : \frac{err_{\mathcal{V}}(\lambda_i) - \min_{\lambda_j} err_{\mathcal{V}}(\lambda_j)}{\min_{\lambda_j} err_{\mathcal{V}}(\lambda_j)} \leq 0.05 \right\} .$$

5. Solve (4.6) on all pixels:

$$\underset{\boldsymbol{Z} \in \mathbb{R}^{K \times T}, \boldsymbol{Z} \geq 0}{\text{minimize}} \quad \frac{1}{2} \left\| \boldsymbol{Y} - \tilde{\boldsymbol{A}} \boldsymbol{Z} \right\|_F^2 + \frac{\lambda^*}{|\mathcal{T}|/P} \alpha \mathbf{1}^T \boldsymbol{Z} \mathbf{1} + \frac{\lambda^*}{|\mathcal{T}|/P} (1 - \alpha) \sum_{k=1}^{K} \|\boldsymbol{z}_{k,\cdot}\|_2 ,$$

   where λ^* is scaled by the percent of pixels in the training set to account for the fact that the sum of squared errors in the loss function is not scaled by the number of pixels.

This process can be done separately for each group of overlapping neurons $\mathcal{N}_1, \ldots, \mathcal{N}_S$ in order to select a different value of λ for each group, or for all groups at once to select a single value of λ. By following steps similar to those described above, λ can alternatively be selected via cross-validation.

C.5 Proof of Lemma 4.6.1

We first prove a result that we will use later.

Lemma C.5.1 *The solution to* $\underset{\boldsymbol{\beta}\geq\mathbf{0}}{\text{minimize}}\ \frac{1}{2}\|\boldsymbol{y}-\boldsymbol{\beta}\|_2^2+\lambda\|\boldsymbol{\beta}\|_2$ *is* $\hat{\boldsymbol{\beta}}=\left(1-\frac{\lambda}{\|(\boldsymbol{y})_+\|_2}\right)_+(\boldsymbol{y})_+$.

Proof Let $\hat{\boldsymbol{\beta}}=\underset{\boldsymbol{\beta}\geq\mathbf{0}}{\text{argmin}}\ \frac{1}{2}\|\boldsymbol{y}-\boldsymbol{\beta}\|_2^2+\lambda\|\boldsymbol{\beta}\|_2$ and $\mathcal{C}=\{i:y_i\geq 0\}$. First, we show $\hat{\boldsymbol{\beta}}_{-\mathcal{C}}=\mathbf{0}$. In anticipation of contradiction, assume there exists j such that $j\notin\mathcal{C}$ and $\hat{\beta}_j>0$. Define $\tilde{\boldsymbol{\beta}}$ as $\tilde{\beta}_i=\begin{cases}\hat{\beta}_i & \text{if } i\neq j\\ 0 & \text{if } i=j\end{cases}$. Then

$$\frac{1}{2}\left\|\boldsymbol{y}-\tilde{\boldsymbol{\beta}}\right\|_2^2+\lambda\left\|\tilde{\boldsymbol{\beta}}\right\|_2<\frac{1}{2}\left\|\boldsymbol{y}-\hat{\boldsymbol{\beta}}\right\|_2^2+\lambda\left\|\hat{\boldsymbol{\beta}}\right\|_2.$$

This is a contradiction, so we conclude that $\hat{\beta}_i=0$ for all $i\notin\mathcal{C}$. It remains to solve

$$\underset{\boldsymbol{\beta}_{\mathcal{C}}\geq\mathbf{0}}{\text{minimize}}\quad\frac{1}{2}\|\boldsymbol{y}_{\mathcal{C}}-\boldsymbol{\beta}_{\mathcal{C}}\|_2^2+\lambda\|\boldsymbol{\beta}_{\mathcal{C}}\|_2. \tag{C.2}$$

By a result in Section 3.1 of Simon et al. [2013], the solution to (C.2) without the non-negativity constraint on $\boldsymbol{\beta}_{\mathcal{C}}$ is $\left(1-\frac{\lambda}{\|\boldsymbol{y}_{\mathcal{C}}\|_2}\right)_+\boldsymbol{y}_{\mathcal{C}}$, which has all non-negative elements. Therefore, it is also the solution to (C.2). ■

We now proceed to prove Lemma 4.6.1.

Proof Our goal is to solve

$$\underset{\boldsymbol{z}\in\mathbb{R}^T,\boldsymbol{z}\geq\mathbf{0}}{\text{minimize}}\quad\frac{1}{2}\left\|\boldsymbol{Y}-\tilde{\boldsymbol{a}}\boldsymbol{z}^T\right\|_F^2+\lambda\alpha\mathbf{1}^T\boldsymbol{z}+\lambda(1-\alpha)\|\boldsymbol{z}\|_2. \tag{C.3}$$

By algebraic manipulation, we can show that

$$\left\|\boldsymbol{Y}-\tilde{\boldsymbol{a}}\boldsymbol{z}^T\right\|_F^2+\lambda\alpha\mathbf{1}^T\boldsymbol{z}=\left\|\frac{\boldsymbol{Y}^T\tilde{\boldsymbol{a}}-\lambda\alpha\mathbf{1}}{\sqrt{\tilde{\boldsymbol{a}}^T\tilde{\boldsymbol{a}}}}-\sqrt{\tilde{\boldsymbol{a}}^T\tilde{\boldsymbol{a}}}\boldsymbol{z}\right\|_2^2+C,$$

where C is a constant that does not depend on $\boldsymbol{z}$. Therefore, the solution to (C.3) is the same as the solution to

$$\underset{\boldsymbol{z}\geq\boldsymbol{0}}{\text{minimize}} \quad \frac{1}{2}\left\|\frac{\boldsymbol{Y}^T\tilde{\boldsymbol{a}}-\lambda\alpha\boldsymbol{1}}{\tilde{\boldsymbol{a}}^T\tilde{\boldsymbol{a}}}-\boldsymbol{z}\right\|_2^2+\frac{\lambda(1-\alpha)}{\tilde{\boldsymbol{a}}^T\tilde{\boldsymbol{a}}}\left\|\boldsymbol{z}\right\|_2. \tag{C.4}$$

We solve (C.4) by applying Lemma C.5.1. ■

C.6 Proof of Lemma 4.6.2

Proof The result follows simply from observing that

$$\begin{aligned}\left\|\boldsymbol{Y}-\tilde{\boldsymbol{A}}\boldsymbol{Z}\right\|_F^2 &= \sum_{s=1}^{S}\left\|\boldsymbol{Y}_{\mathcal{M}(\mathcal{N}_s),\cdot}-\tilde{\boldsymbol{A}}_{\mathcal{M}(\mathcal{N}_s),\cdot}\boldsymbol{Z}\right\|_F^2\\ &= \sum_{s=1}^{S}\left\|\boldsymbol{Y}_{\mathcal{M}(\mathcal{N}_s),\cdot}-\sum_{s'=1}^{S}\tilde{\boldsymbol{A}}_{\mathcal{M}(\mathcal{N}_s),\mathcal{N}_{s'}}\boldsymbol{Z}_{\mathcal{N}_{s'},\cdot}\right\|_F^2\\ &= \sum_{s=1}^{S}\left\|\boldsymbol{Y}_{\mathcal{M}(\mathcal{N}_s),\cdot}-\tilde{\boldsymbol{A}}_{\mathcal{M}(\mathcal{N}_s),\mathcal{N}_s}\boldsymbol{Z}_{\mathcal{N}_s,\cdot}\right\|_F^2.\end{aligned}$$

The last equality follows from the condition of the lemma, which guarantees that $\tilde{\boldsymbol{A}}_{\mathcal{M}(\mathcal{N}_s),\mathcal{N}_{s'}}=0$ for all $s\neq s'$. ■

C.7 Details of Step 2(b) of Algorithm 4

Let $f\left(\boldsymbol{Z}_{\mathcal{N}_s,\cdot}\right)=\frac{1}{2}\left\|\boldsymbol{Y}_{\mathcal{M}(\mathcal{N}_s),\cdot}-\tilde{\boldsymbol{A}}_{\mathcal{M}(\mathcal{N}_s),\mathcal{N}_s}\boldsymbol{Z}_{\mathcal{N}_s,\cdot}\right\|_F^2+\lambda\alpha\boldsymbol{1}^T\boldsymbol{Z}_{\mathcal{N}_s,\cdot}\boldsymbol{1}$, which is the differentiable part of the objective function in (4.9), and let $g\left(\boldsymbol{Z}_{\mathcal{N}_s,\cdot}\right)=\lambda(1-\alpha)\sum_{n\in\mathcal{N}_s}\left\|\boldsymbol{z}_{n,\cdot}\right\|_2$, the non-differentiable part.

Generalized gradient descent [Beck and Teboulle, 2009, Parikh and Boyd, 2014] is a majorization-minimization scheme. First, we find a quadratic approximation to $f\left(\boldsymbol{Z}_{\mathcal{N}_s,\cdot}\right)$ centered at our previous estimate for $\boldsymbol{Z}_{\mathcal{N}_s,\cdot}$, $\boldsymbol{Z}^0_{\mathcal{N}_s,\cdot}$, that majorizes $f\left(\boldsymbol{Z}_{\mathcal{N}_s,\cdot}\right)$. That is,

$$f\left(\boldsymbol{Z}_{\mathcal{N}_s,\cdot}\right)\leq f\left(\boldsymbol{Z}^0_{\mathcal{N}_s,\cdot}\right)+\operatorname{Tr}\left[\left(\boldsymbol{Z}_{\mathcal{N}_s,\cdot}-\boldsymbol{Z}^0_{\mathcal{N}_s,\cdot}\right)^T\nabla f\left(\boldsymbol{Z}^0_{\mathcal{N}_s,\cdot}\right)\right]+\frac{1}{2t}\left\|\boldsymbol{Z}_{\mathcal{N}_s,\cdot}-\boldsymbol{Z}^0_{\mathcal{N}_s,\cdot}\right\|_F^2,$$

where t is the step size such that $\nabla^2 f(\cdot) \preceq \frac{1}{t}\boldsymbol{I}$. After completing the square, we can see that minimizing the quadratic approximation to $f\left(\boldsymbol{Z}_{\mathcal{N}_s,\cdot}\right)$ gives the same solution as solving

$$\underset{\boldsymbol{Z}_{\mathcal{N}_s,\cdot}}{\text{minimize}} \quad \frac{1}{2t}\left\|\boldsymbol{Z}_{\mathcal{N}_s,\cdot} - \left(\boldsymbol{Z}^0_{\mathcal{N}_s,\cdot} - t\nabla f\left(\boldsymbol{Z}^0_{\mathcal{N}_s,\cdot}\right)\right)\right\|_F^2.$$

Thus we perform this minimization with $g\left(\boldsymbol{Z}_{\mathcal{N}_s,\cdot}\right)$ added to the objective function, which gives the proximal problem

$$\underset{\boldsymbol{Z}_{\mathcal{N}_s,\cdot}\geq\boldsymbol{0}}{\text{minimize}} \quad \frac{1}{2}\left\|\boldsymbol{Z}_{\mathcal{N}_s,\cdot} - \tilde{\boldsymbol{Y}}_{\mathcal{M}(\mathcal{N}_s),\cdot}\right\|_F^2 + \lambda(1-\alpha)t\sum_{n\in\mathcal{N}_s}\left\|\boldsymbol{z}_{n,\cdot}\right\|_2, \tag{C.5}$$

where $\tilde{\boldsymbol{Y}}_{\mathcal{M}(\mathcal{N}_s),\cdot} = \boldsymbol{Z}^0_{\mathcal{N}_s,\cdot} - t\left(-(\tilde{\boldsymbol{A}}_{\mathcal{M}(\mathcal{N}_s),\mathcal{N}_s})^T\left(\boldsymbol{Y}_{\mathcal{M}(\mathcal{N}_s),\cdot} - \tilde{\boldsymbol{A}}_{\mathcal{M}(\mathcal{N}_s),\mathcal{N}_s}\boldsymbol{Z}^0_{\mathcal{N}_s,\cdot}\right) + \lambda\alpha\boldsymbol{1}\boldsymbol{1}^T\right)$. The minimization in (C.5) is separable in $\boldsymbol{z}_{n,\cdot}$, so for $n\in\mathcal{N}_s$, we solve

$$\underset{\boldsymbol{z}_{n,\cdot}\geq\boldsymbol{0}}{\text{minimize}} \quad \frac{1}{2}\left\|\boldsymbol{z}_{n,\cdot} - \tilde{\boldsymbol{y}}_{n,\cdot}\right\|_2^2 + \lambda(1-\alpha)t\left\|\boldsymbol{z}_{n,\cdot}\right\|_2. \tag{C.6}$$

By Lemma C.5.1 in Appendix C.5, the solution to (C.6) is $\hat{\boldsymbol{z}}_{n,\cdot} = \left(1 - \frac{\lambda(1-\alpha)t}{\left\|(\tilde{\boldsymbol{y}}_{n,\cdot})_+\right\|_2}\right)_+(\tilde{\boldsymbol{y}}_{n,\cdot})_+$.

It only remains to derive a suitable step size t so that $\nabla^2 f(\cdot) = (\tilde{\boldsymbol{A}}_{\mathcal{M}(\mathcal{N}_s),\mathcal{N}_s})^T\tilde{\boldsymbol{A}}_{\mathcal{M}(\mathcal{N}_s),\mathcal{N}_s} \preceq \frac{1}{t}\boldsymbol{I}$. A sufficient condition for $\frac{1}{t}\boldsymbol{I} - (\tilde{\boldsymbol{A}}_{\mathcal{M}(\mathcal{N}_s),\mathcal{N}_s})^T\tilde{\boldsymbol{A}}_{\mathcal{M}(\mathcal{N}_s),\mathcal{N}_s}$ to be positive semi-definite is that $\frac{1}{t}\boldsymbol{I} - (\tilde{\boldsymbol{A}}_{\mathcal{M}(\mathcal{N}_s),\mathcal{N}_s})^T\tilde{\boldsymbol{A}}_{\mathcal{M}(\mathcal{N}_s),\mathcal{N}_s}$ be diagonally dominant. That is,

$$\frac{1}{t} - (\tilde{\boldsymbol{a}}_{\mathcal{M}(\mathcal{N}_s),n})^T\tilde{\boldsymbol{a}}_{\mathcal{M}(\mathcal{N}_s),n} \geq \sum_{j\in\mathcal{N}_s, j\neq n}(\tilde{\boldsymbol{a}}_{\mathcal{M}(\mathcal{N}_s),j})^T\tilde{\boldsymbol{a}}_{\mathcal{M}(\mathcal{N}_s),n}$$

for all $n\in\mathcal{N}_s$. Thus we choose $t = (\max_{n\in\mathcal{N}_s}\sum_{j\in\mathcal{N}_s}(\tilde{\boldsymbol{a}}_{\mathcal{M}(\mathcal{N}_s),j})^T\tilde{\boldsymbol{a}}_{\mathcal{M}(\mathcal{N}_s),n})^{-1}$.

C.8 Proof of Lemma 4.6.3

Proof Since $\boldsymbol{a}_{\cdot,1}$ does not overlap any other spatial components (i.e., $(\boldsymbol{a}_{\cdot,1})^T\boldsymbol{a}_{\cdot,k} = 0$ for $k = 2,\ldots,K$), we know by the results in Lemmas 4.6.1 and 4.6.2 that

$$\hat{\boldsymbol{z}}_{1,\cdot} = \left(1 - \frac{\lambda(1-\alpha)}{\|(\boldsymbol{Y}^T\tilde{\boldsymbol{a}}_{\cdot,1} - \lambda\alpha\boldsymbol{1})_+\|_2}\right)_+\left(\frac{\boldsymbol{Y}^T\tilde{\boldsymbol{a}}_{\cdot,1} - \lambda\alpha\boldsymbol{1}}{(\tilde{\boldsymbol{a}}_{\cdot,1})^T\tilde{\boldsymbol{a}}_{\cdot,1}}\right)_+.$$

Note that $\boldsymbol{Y}^T\tilde{\boldsymbol{a}}_{\cdot,1} = (\boldsymbol{Z}^*)^T\boldsymbol{A}^T\boldsymbol{a}_{\cdot,1}/\|\boldsymbol{a}_{\cdot,1}\|_2^2$, so $\boldsymbol{Y}^T\tilde{\boldsymbol{a}}_{\cdot,1} = \boldsymbol{z}^*_{1,\cdot}$ since $(\boldsymbol{a}_{\cdot,1})^T\boldsymbol{a}_{\cdot,k} = 0$ for $k = 2,\ldots,K$. Thus $\hat{\boldsymbol{z}}_{1,\cdot} = \boldsymbol{0}$ if and only if $\lambda(1-\alpha) \geq \|(\boldsymbol{z}^*_{1,\cdot} - \lambda\alpha\boldsymbol{1})_+\|_2$. Similarly, $\hat{\boldsymbol{z}}_{2,\cdot} = \boldsymbol{0}$ if

and only if $\lambda(1-\alpha) \geq \|(\boldsymbol{z}^*_{2,\cdot} - \lambda\alpha\mathbf{1})_+\|_2$. Thus, it remains to show that $\|(\boldsymbol{z}^*_{1,\cdot} - \lambda\alpha\mathbf{1})_+\|_2 = \|(\boldsymbol{z}^*_{2,\cdot} - \lambda\alpha\mathbf{1})_+\|_2$. Using the properties of permutation matrices that $\mathbf{1} = \boldsymbol{P}\mathbf{1}$ and $\boldsymbol{P}^T\boldsymbol{P} = \boldsymbol{I}$, we see

$$\begin{aligned}\|(\boldsymbol{z}^*_{1,\cdot} - \lambda\alpha\mathbf{1})_+\|_2 &= \|(\boldsymbol{P}\boldsymbol{z}^*_{2,\cdot} - \lambda\alpha\boldsymbol{P}\mathbf{1})_+\|_2 \\ &= \|\boldsymbol{P}(\boldsymbol{z}^*_{2,\cdot} - \lambda\alpha\mathbf{1})_+\|_2 \\ &= \|(\boldsymbol{z}^*_{2,\cdot} - \lambda\alpha\mathbf{1})_+\|_2,\end{aligned}$$

and therefore, $\hat{\boldsymbol{z}}_{1,\cdot} = \mathbf{0}$ if and only if $\hat{\boldsymbol{z}}_{2,\cdot} = \mathbf{0}$. ■

C.9 Proof of Lemma 4.6.4

Proof Recall that solving (4.6) gives the same solution as solving (4.8). Thus we consider when $\hat{\boldsymbol{Z}}_{\mathcal{N}_s,\cdot}$, the solution to (4.8), is sparse for $s = 1, \ldots, S$. If $|\mathcal{N}_s| = 1$, we see from (4.10) that $\hat{\boldsymbol{z}}_{\mathcal{N}_s,\cdot} = \mathbf{0}$ if and only if

$$\lambda(1-\alpha) \geq \left\| \left((\boldsymbol{Y}_{\mathcal{M}(\mathcal{N}_s),\cdot})^T \tilde{\boldsymbol{a}}_{\mathcal{M}(\mathcal{N}_s),\mathcal{N}_s} - \lambda\alpha\mathbf{1} \right)_+ \right\|_2 . \tag{C.7}$$

Recall that if $|\mathcal{N}_s| > 1$, we iteratively solve for $\hat{\boldsymbol{Z}}_{\mathcal{N}_s,\cdot}$ using Step 2(b) of Algorithm 4. We initialize at the sparse solution $\boldsymbol{Z}^{(0)}_{\mathcal{N}_s,\cdot} = \mathbf{0}$ and thus for $n \in \mathcal{N}_s$

$$\boldsymbol{z}^{(1)}_{n,\cdot} = \left(1 - \frac{\lambda(1-\alpha)t}{\left\| (\tilde{\boldsymbol{y}}_{n,\cdot})_+ \right\|_2} \right)_+ (\tilde{\boldsymbol{y}}_{n,\cdot})_+ ,$$

where $\tilde{\boldsymbol{Y}}_{\mathcal{N}_s,\cdot} = t(\tilde{\boldsymbol{A}}_{\mathcal{M}(\mathcal{N}_s),\mathcal{N}_s})^T \boldsymbol{Y}_{\mathcal{N}_s,\cdot} - t\lambda\alpha\mathbf{1}\mathbf{1}^T$. We will have $\hat{\boldsymbol{Z}}_{\mathcal{N}_s,\cdot} = \mathbf{0}$ if $\boldsymbol{z}^{(1)}_{n,\cdot} = \mathbf{0}$ for all $n \in \mathcal{N}_s$. Note that $\boldsymbol{z}^{(1)}_{n,\cdot} = \mathbf{0}$ if

$$\lambda(1-\alpha)t \geq \left\| \left(t \left[(\tilde{\boldsymbol{A}}_{\mathcal{M}(\mathcal{N}_s),\mathcal{N}_s})^T \boldsymbol{Y}_{\mathcal{N}_s,\cdot} \right]_{n,\cdot} - t\lambda\alpha\mathbf{1} \right)_+ \right\|_2 . \tag{C.8}$$

By algebraic manipulation, the sparsity conditions given in (C.7) and (C.8) can be shown to be equivalent to the condition given in Lemma 4.6.4. Alternatively, this lemma's result also follows from inspection of the optimality condition for (4.6). ■

C.10 Proof of Corollary 4.6.5

Proof The sufficient condition given in Corollary 4.6.5 follows from noting that (C.8) is satisfied if $\lambda(1-\alpha) \geq \left\| \left(\left[\tilde{\boldsymbol{A}}^T \boldsymbol{Y} \right]_{k,\cdot} \right)_+ \right\|_2$ or if $\lambda\alpha \geq \left(\left[\tilde{\boldsymbol{A}}^T \boldsymbol{Y} \right]_{k,l} \right)_+$ for $l = 1, \ldots, T$. Thus, when at least one of these two conditions is satisfied for all $k = 1, \ldots, K$, then the solution to (4.6) will be sparse. ■