Sets as Measures: Optimization and Machine Learning

by

Nicholas Boyd

A dissertation submitted in partial satisfaction of the

requirements for the degree of

Doctor of Philosophy

in

Statistics

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Professor Benjamin Recht, Co-chair
Professor Michael Jordan, Co-chair
Professor Peter Bartlett
Professor Yi-Ren Ng

Spring 2018

Sets as Measures: Optimization and Machine Learning

Abstract

Sets as Measures: Optimization and Machine Learning

by

Nicholas Boyd

Doctor of Philosophy in Statistics

University of California, Berkeley

Professor Benjamin Recht, Co-chair

Professor Michael Jordan, Co-chair

The purpose of this thesis is to address the following simple question:

How do we design efficient algorithms to solve optimization or machine learning problems where the decision variable (or target label) is a set *of unknown cardinality?*

In this thesis we show that, in some cases, optimization and machine learning algorithms designed to work with single vectors can be directly applied to problems involving sets. We do this by invoking a classical trick: we lift sets to elements of a vector space, namely an infinite-dimensional space of measures. While this idea has been explored extensively in theoretical analysis, we show that it also generates efficient practical algorithms.

Contents

List of Figures

List of Tables

Acknowledgments

I would like to thank my advisors, Benjamin Recht and Michael Jordan, for their support and guidance. I am also very grateful to the Fannie and John Hertz Foundation, along with Google, for providing financial support during the majority of my graduate studies. I would also like to thank Dan Klein, Laurent El Ghaoui, and Claire Tomlin for guidance and inspiration during my many years at Berkeley.

I must also thank my family — Anna, Stephen, and Nora — for their invaluable support and love. I still find it hard to believe that I am related to such wonderful people. My father, Stephen, should also be thanked separately for the innumerable hours spent serving as my de facto third advisor.

My coauthors, and in particular Geoff and Eric, deserve special thanks for putting up with me and keeping me on track, and for making research vastly more fun. I'd also like to thank Trevor Hastie for many valuable discussions about the work in Chapter 4. Finally, I'd like to thank Anna Thompson for help with diagrams in Chapter 5, and for invaluable support.

The best part of my time at Berkeley has been outside of work, with the amazing people I will be forever grateful to have as friends. I am indebted to too many to list here, without whom I would never have finished graduate school, and, more importantly, who made my time in Berkeley some of the best in my life.

Chapter 1

The Basic Idea

1.1 Introduction

The purpose of this thesis is to address the following simple question:

How do we design efficient algorithms to solve optimization or machine learning problems where the decision variable (or target label) is a set *of unknown cardinality?*

Optimization and machine learning have proved remarkably successful in applications requiring the choice of single vectors. Some tasks, in particular many inverse problems, call for the design, or estimation, of *sets* of objects. When the size of these sets is a priori unknown, directly applying optimization or machine learning techniques designed for single vectors appears difficult. The work in this thesis shows that a very old idea for transforming sets into elements of a vector space (namely, a space of measures), a common trick in theoretical analysis, generates effective *practical* algorithms.

Essentially everything in this thesis follows from the following simple observation: a finite set of objects $\{\theta_1, \ldots, \theta_n\} \subset \Theta$ can be identified with the *measure* $\sum_{i=1}^n \delta_{\theta_i}$ under the following correspondence:

$$\{\theta_1, \ldots, \theta_n\} \leftrightarrow \sum_{i=1}^{n} \delta_{\theta_i}. \tag{1.1}$$

Here δ_θ is a point mass at θ, defined by

$$\delta_\theta(A) = \begin{cases} 1 & \text{if } \theta \in A \\ 0 & \text{if } \theta \notin A. \end{cases} \tag{1.2}$$

This measure is an element of a vector space in which we can directly apply many standard optimization and machine learning algorithms.

Chapter 1 explains this simple idea and sets up the mathematical framework for working with measures.

Chapter 2 describes a class of infinite-dimensional convex optimization problems with set-valued decision variables that we term sparse linear inverse problems (SLIP), and establishes a strong connection between these problems and the well-known framework of atomic norms.

Chapter 3 shows that a classical convex optimization algorithm can be directly applied to sparse linear inverse problems, and introduces a simple extension with vastly superior practical performance: the Alternating Descent Conditional Gradient Method (ADCG). Finally, we demonstrate the efficacy of ADCG in three application areas. Chapters 2 and 3 are based on the paper:

N. Boyd, G. Schiebinger, and B. Recht. "The Alternating Descent Conditional Gradient Method for Sparse Inverse Problems". In: *SIAM Journal on Optimization* 27.2 (2017), pp. 616–639.

Chapter 4 explores the application of these ideas to a classical statistical estimation problem: fitting adaptive splines and adaptive spline generalized additive models to data. We also show a simple extension of the standard adaptive spline setup that allows for feature selection in generalized additive models; this modification requires a slight extension of ADCG to handle simple constraints. Chapter 4 is based on the paper:

N. Boyd, T. Hastie, S. Boyd, B. Recht, and M. Jordan. "Saturating Splines and Feature Selection". In: *Journal of Machine Learning Research* 17.178 (2017).

In Chapter 5 we revisit the single molecule localization microscopy application from Chapter 3 and show how to train neural networks that output sets of points. To do so we treat the output of the network, a collection of points, as a measure and use techniques from statistics, namely embeddings of measures into reproducing kernel Hilbert spaces, to define a loss function. The material in Chapter 5 is from the paper:

N. Boyd, E. Jonas, H. Babcock, and B. Recht. "DeepLoco: Fast 3D Localization Microscopy Using Neural Networks". In: *bioRxiv* (2018).

1.2 Common notation

In this section we introduce common notation. We first informally describe the setup before giving rigorous definitions. While we assume a basic familiarity with measure theory (for a review see [42]), a reader unfamiliar with the subject can follow along without too much trouble. We conclude with a brief discussion of why we consider measures the natural choice.

The objects of interest in this thesis are sets of points of the form $\{\theta_1, \ldots, \theta_n\} \subset \Theta$. We'll also often deal with *weighted* sets of points, of the form $\{(w_1, \theta_1), \ldots, (w_n, \theta_n)\} \subset \mathbf{R} \times \Theta$.

We'll often talk about unweighted sets of points (like γ) as weighted collections, in which case we'll take each w_i to be one. We'll delay discussion of the set Θ — which we'll often call the *parameter space* — until later, but a good example to keep in mind would be the unit box in $\mathbf{R}^p$ for p small: $\{x : \|x\|_\infty \leq 1\}$.

The main idea in this thesis is to make use of a bijection between weighted collections of (unique) objects in Θ and *finitely-supported* atomic signed measures on Θ. The bijection associates a weighted collection $\gamma = \{(w_1, \theta_1), \ldots, (w_n, \theta_n)\}$ with the measure

$$\mathcal{M}(\gamma) = \sum_{i=1}^{n} w_i \delta_{\theta_i}. \tag{1.3}$$

Again, δ_θ is a point-mass supported on θ. Similarly, if μ is a finitely-supported atomic measure on Θ, $\mathcal{M}^{-1}(\mu)$ is a well-defined weighted set of points. We will call such measures *sparse*.

We'll often move back and forth between sparse measures and weighted collections, so we highlight two (very simple) identities we will use frequently. In the following, $\mu = \mathcal{M}(\gamma)$. If ϕ is a (bounded, continuous) function from Θ into a vector space there is a natural lift to a linear operator Φ on atomic measures (indeed, it is well defined for all Borel measures of finite total variation),

$$\Phi\mu = \int \phi(\theta) d\mu(\theta) = \sum_i w_i \phi(\theta_i). \tag{1.4}$$

We stress here that while Φ is *always* a linear operator, ϕ need not be linear: indeed, in all examples in this thesis it is not. The total variation of the measure μ ($|\mu|(\Theta)$) is exactly the ℓ_1 norm of the weights w considered as a vector:

$$\|\mu\|_{\mathrm{TV}} = \|w\|_1 = \sum_i |w_i|. \tag{1.5}$$

In light of (1.5), we'll often use $\|\mu\|_1$ interchangeably with $\|\mu\|_{\mathrm{TV}}$.

Formal requirements

Throughout we'll assume Θ is a compact subset of a differentiable manifold equipped with the Borel σ-algebra. In this case, the space of measures of finite total variation on Θ is a Banach space. Furthermore, we'll assume ϕ (or any function on Θ) is bounded and continuous.

Why measures?

Almost all of the measures in this thesis will be atomic and supported on finitely many points (sparse), so they may be represented as weighted collections, in which case (1.4) and (1.5) are (essentially) the only identities a reader will need.

This observation begs the question: why deal with measures at all? Indeed, all of our algorithms will manipulate finite sets of weighted points, not general measures; can we not deal directly with these objects mathematically? The answer is that we could do this with a bit of additional work, but we believe that measures are much more familiar to most readers than other options. Optimization algorithms typically operate in (at least) a Banach space — the space of weighted collections is not complete. In fact, depending on the choice of topology for this space — which can be identified with the space of formal linear combinations of elements from Θ — we might end up with a (relatively) odd space unfamiliar to most readers, or a space essentially identical to the space of measures we've chosen. For this reason, we feel that working with measures is a good compromise between familiarity (to most readers) and mathematical rigor. With that said, we remind the reader that this distinction is mostly theoretical: the algorithms we present manipulate sparse measures, which may be identified with weighted collections.

Chapter 2

Sparse Linear Inverse Problems

In this chapter we introduce a particular class of convex optimization problems on measures which arise naturally when the decision variable is a weighted collection. This optimization problem has a long history, and has appeared recently in theoretical work as a natural generalization of compressed sensing. We discuss connections to conventional compressed sensing and atomic norm problems. To provide motivation we describe several problems that can be formulated using this framework. Finally we provide a simple theoretical result that ensures we can always find a solution to a SLIP that is atomic and finitely supported.

2.1 Introduction

A ubiquitous prior in modern statistical signal processing asserts that an observed signal is the noisy observation of a few weighted sources. In other words, compared to the entire dictionary of possible sources, the set of sources actually present is *sparse*. In the most abstract formulation of this prior, each source is chosen from a non-parametric dictionary, but in many cases of practical interest the sources are parameterized. Hence, solving the inverse problem amounts to finding a collection of a few parameters and weights that adequately explains the observed signal.

As an example, consider the idealized task of identifying the aircraft that explain an observed radar signal. The sources are the aircraft themselves, and each is parameterized by, perhaps, its position and velocity relative to the radar detector. The inverse problem is to recover the number of aircraft present, along with each of their parameters.

As discussed in Chapter 1, any collection of weighted sources can be represented as a measure on the parameter space: each source corresponds to a single point mass at its corresponding parameter value. We will call atomic measures supported on finitely many points sparse measures. When the parameter spaces are infinite—for example the set of all velocities and positions of aircraft — the space of bounded measures over such parameters is infinite-dimensional. This means that optimization problems searching for parsimonious explanations of the observed signal must operate over an infinite-dimensional space.

Many alternative formulations of the sparse inverse problem have been proposed to avoid the infinite-dimensional optimization required in the sparse measure setup. The most canonical and widely applicable approach is to form a discrete grid over the parameter space and restrict the search to measures supported on the grid. This restriction produces a finite-dimensional optimization problem [118, 78, 113]. In certain special cases, the infinite-dimensional optimization problem over measures can be reduced to a problem of moment estimation, and spectral techniques or semidefinite programming can be employed [50, 90, 112, 19]. More recently, in light of much of the work on compressed sensing and its generalizations, another proposal operates on atomic norms over data [21], opening other algorithmic possibilities.

While these finite-dimensional formulations are appealing, they all essentially treat the space of sources as an unstructured set, ignoring natural structure (such as differentiability) present in many applications. All three of these techniques have their individual drawbacks, as well. Gridding only works for very small parameter spaces, and introduces artifacts that often require heuristic post-processing [113]. Moment methods have limited applicability, are typically computationally expensive, and, moreover, are sensitive to noise and estimates of the number of sources. Finally, atomic norm techniques do not recover the parameters of the underlying signal, and as such are more naturally applied to denoising problems. For a much more in-depth discussion of atomic norms, see §2.3.

We argue that all of these issues can be alleviated by returning to the original formulation of the estimation problem as an optimization problem over a space of measures. Working with measures explicitly exposes the underlying parameter space, which allows us to consider algorithms that make local moves within parameter space. Chapters 3 and 4 demonstrate that operating on an infinite-dimensional space of measures is not only feasible algorithmically, but that the resulting algorithms outperform techniques based on gridding or moments on a variety of real-world signal processing tasks. In Chapter 3 we formalize a general approach to solving parametric sparse inverse problems via the conditional gradient method (CGM), also know as the Frank-Wolfe algorithm. We show how to augment the classical CGM with nonconvex local search exploiting structure in the parameter space. This hybrid scheme, which we call the alternating descent conditional gradient method (ADCG), enjoys both the rapid local convergence of nonconvex programming algorithms and the stability and global convergence guarantees associated with convex optimization.

Mathematical setup

In this subsection we formalize the sparse inverse problem as an optimization problem over measures and discuss a convex heuristic.

We assume the existence of an underlying collection of objects, called sources. Each source has a scalar weight w, and a parameter $\theta \in \Theta$. We require the parameter space be measurable (that is, equipped with a σ-algebra) and amenable to local, derivative-based optimization; formally, we'll want Θ to be a compact subset of a differentiable manifold. Some examples to keep in mind would be Θ a compact subset of $\mathbf{R}^p$ for some small p, or the sphere $\mathcal{S}^p$ considered as a differentiable manifold. An element θ of the parameter space Θ may describe, for instance, the position, orientation, and polarization of a source. The weight w may encode the intensity of a source, or the distance of a source from the observation device. Our goal is to recover the number of sources present, along with their individual weights and parameters. We do not observe the sources directly, but instead are given a single, noisy observation.

The observation model we use is completely specified by a function $\phi : \Theta \to \mathbf{R}^d$, which gives the d-dimensional observation of a single, unit-weight source parameterized by a point in Θ. A single source with parameter θ and weight w generates the observation $w\phi(\theta) \in \mathbf{R}^d$: that is, the observation of a lone source is homogeneous of degree one in its weight. Finally, we assume that the observation generated by a weighted collection of sources is additive. In other words, the (noise-free) observation of a weighted collection is generated by the mapping

$$\gamma = \{(w_i, \theta_i)\}_{i=1}^{K} \mapsto \sum_{i=1}^{K} w_i \phi(\theta_i) \in \mathbf{R}^d. \tag{2.1}$$

We refer to the collection $\gamma = \{(w_i, \theta_i)\}_{i=1}^{K}$ as the *signal parameters*, and the vector $\sum_{i=1}^{K} w_i \phi(\theta_i) \in \mathbf{R}^d$ as the noise-free *observation*. We require that ϕ be bounded: $\|\phi(\theta)\|_2^2 \leq 1$ for all θ, and further that ϕ be *differentiable* in θ. Finally, let us emphasize that we make no further assumptions about ϕ: in particular it does *not* need to be linear. It's worth noting here that with K and $\theta_1, \ldots, \theta_K$ held fixed, (2.1) is *linear* in the weights $w_1, \ldots, w_K$.

Our goal is to recover the true weighted collection of sources, $\{(\tilde{w}_i, \tilde{\theta}_i)\}_{i=1}^{\tilde{K}}$, from a single noisy observation:

$$y \simeq \sum_{i=1}^{\tilde{K}} \tilde{w}_i \phi(\tilde{\theta}_i).$$

We emphasize that the goal is *not* to denoise the vector y: that is we are not satisfied with recovering the noise-free observation (i.e. the vector $\sum_{i=1}^{\tilde{K}} \tilde{w}_i \phi(\tilde{\theta}_i)) \in \mathbf{R}^d$), but rather we require an estimate of the true signal parameters $\{(\tilde{w}_i, \tilde{\theta}_i)\}$. This is in stark contrast with the atomic norm case discussed in §2.3.

One approach would be to attempt to minimize a (differentiable) convex loss, ℓ, that describes the likelihood of observing the vector y given the expected output for an estimated collection of sources:

$$\begin{array}{ll} \underset{w,\theta,K}{\text{minimize}} & \ell\left(\sum_{i=1}^{K} w_i \phi(\theta_i)\right) \\ \text{subject to} & K \leq N. \end{array} \tag{2.2}$$

Here N is a posited upper bound on the number of sources. For example, if ℓ is the negative log-likelihood of the noise process, problem (2.2) corresponds to maximum-likelihood estimation of the true sources. Unfortunately, (2.2) is nonconvex in the variables w, θ, and K. As such, algorithms designed to solve this problem are hard to reason about and come with few guarantees. Also, in practice they often suffer from sensitivity to initialization. In this thesis, we *lift* the problem to a space of (signed) measures on Θ; this lifting allows us to apply a natural heuristic to devise a convex surrogate for problem (2.2).

We can encode an arbitrary, weighted collection of sources γ as a sparse measure μ on Θ, with mass w_i at point θ_i: $\mu = \sum_{i=1}^{K} w_i \delta_{\theta_i} = \mathcal{M}(\gamma)$. As a consequence of the additivity and homogeneity in our observation model, the total observation of a collection of sources encoded in the measure μ is a linear function Φ of μ:

$$\Phi\mu = \int \phi(\theta) d\mu(\theta).$$

We call Φ the *forward operator*. For atomic measures of the form $\mu = \sum_{i=1}^{n} w_i \delta_{\theta_i}$, this clearly agrees with (2.1); but it is defined for more general measures on Θ.

We now introduce the sparse inverse problem as an optimization problem over the Banach space of bounded, signed measures on the measurable space Θ equipped with the total variation norm [34]. To reiterate, our goal is to recover μ_{true} from an observation

$$y \simeq \Phi\mu_{\text{true}}$$

corrupted by noise. Recovering the signal parameters without any prior information is, in most interesting problems, impossible; the operator Φ is almost never injective. However, in a sparse inverse problem we have the prior belief that the number of sources present, while still unknown, is small. That is, we assume that μ_{true} is an atomic measure supported on very few points.

To make the connection to compressed sensing clear, we refer to such measures as *sparse* measures. Note that while we are using the language of *recovery* or *estimation* in this section, the optimization problem we introduce is also applicable in cases where these may not be a true measure underlying the observation model. In §2.2 we give several examples that are not recovery problems.

We estimate the signal parameters encoded in μ_{true} by minimizing the loss ℓ applied to $\Phi\mu$:

$$\begin{array}{ll} \text{minimize} & \ell\left(\Phi\mu\right) \\ \text{subject to} & |\text{supp}(\mu)| \leq N, \end{array} \tag{2.3}$$

where the optimization is over the Banach space of signed measures (on Θ) equipped with the total variation norm. Here we constrain the cardinality of the support of the measure μ by N, a posited upper bound on the size of the support of the true measure μ_{true}. Although here and elsewhere in this thesis we place no constraint on the sign of w (and hence μ), all of our discussion and algorithms can be easily extended to the non-negative case by adding the requirement that μ be a non-negative measure.

While the objective function in (2.3) is convex, the constraint on the support of μ is nonconvex. A common heuristic in this situation is to replace the nonconvex constraint with a convex surrogate. The standard surrogate for a cardinality constraint on a measure is a constraint on the total variation [19]. This substitution results in the following convex approximation to (2.3):

$$\begin{array}{ll} \text{minimize} & \ell\left(\Phi\mu\right) \\ \text{subject to} & |\mu|(\Theta) \leq \tau. \end{array} \tag{2.4}$$

Here $\tau > 0$ is a parameter that controls the total mass of μ and empirically controls the cardinality of solutions to (2.4). While problem (2.4) is convex, it is over an infinite-dimensional space, and it is not possible to represent an arbitrary measure in a computer. A priori, an approximate solution to (2.4) may have arbitrarily large support, though we prove in §3.4 that we can always find solutions supported on at most $d+1$ points. In practice, however, we are interested in approximate solutions of (2.4) supported on far, far fewer than $d+1$ points.

In Chapter 3, we propose an algorithm to solve (2.4) in the case where Θ is amenable to local, derivative based optimization — in other words, the case where ϕ is differentiable. Our algorithm is based on a variant of the conditional gradient method that takes advantage of the differentiable nature of ϕ, and is guaranteed to produce approximate solutions with bounded support.

Relationship to the lasso. Readers familiar with techniques for estimating sparse vectors may recognize (2.4) as a continuous analogue of the standard lasso. In particular, the standard lasso is an instance of (2.4) with $\ell(o) = \frac{1}{2}\|o - y\|_2^2$ and $\Theta = \{1, \ldots, k\}$. In that case, a measure over Θ can be represented as a vector v in $\mathbf{R}^k$ and the forward operator Φ as a matrix in $\mathbf{R}^{d \times k}$. The total variation of the measure v is then simply $\sum_i |v_i| = \|v\|_1$. We caution the reader that this discrete setup is substantially different as the parameter space has no differential structure. However, to make the connection to the finite dimensional case clear, we will use the notation $\|\mu\|_1$ to refer to the total variation of the measure μ.

A note on measures. While the optimization problem (2.4) has as the decision variable a general measure μ on Θ, due to the nature of the algorithms we discuss we will only ever deal with sparse measures. Sparse measures, that is measures of the form $\mu = \sum_{i=1}^{K} w_i \phi(\theta_i)$, can always be thought of as simple sets of weighted parameters: $\{(w_i, \theta_i)\}_{i=1}^{K}$, or, equivalently a pair of vectors $w \in \mathbf{R}^K$, $\vec{\theta} \in \Theta^K$. Indeed, we will often move back and forth between these equivalent representations. The reader unfamiliar with measure theory can think of the algorithm as operating on this alternative representation directly. Two important identities to keep in mind when dealing with sparse measures are (2.1) and

$$\|\mu\|_1 = \|w\|_1.$$

For a review of basic measure theory, see [42].

2.2 Example applications

Many practical problems can be formulated as instances of (2.4). In this section we briefly outline a few examples to motivate our study of this problem.

Single molecule localization microscopy. The diffraction of light imposes a physical limit on the resolution of optical images. The goal of single molecule localization microscopy (SMLM) superresolution imaging is to remove the blur induced by diffraction as well as the effects of pixelization and noise. For images composed of a collection of point sources of light, this can be posed as a sparse inverse problem as follows. The parameters $\theta_1, \ldots, \theta_K$ denote the locations of K point sources (in $[0,1]^2$ or $[0,1]^3$), and $w_i > 0$ denotes the intensity, or brightness, of the ith source. The image of the ith source is given by $w_i\phi(\theta_i)$, where ϕ is the pixelated point spread function of the imaging apparatus. One option for the loss function is $\ell(o) = \frac{1}{2}\|o - y\|^2$, where y is the observed image. By solving a version of (2.4) it is sometimes possible to localize the point sources better than the diffraction limit—even with extreme pixelization. Astronomers use this framework to deconvolve images of stars to angular resolution below the Rayleigh limit [92]. In biology this tool has revolutionized imaging of sub-cellular features [38, 101]. A variant of this framework allows imaging through scattering media [76]. In §3.3, we show that our algorithm improves upon the current state of the art for localizing point sources in a fluorescence microscopy challenge dataset.

Linear system identification. Linear time-invariant (LTI) dynamical systems are used to model many physical systems. Such a model describes the evolution of an output $y_t \in \mathbb{R}$ based on the input $u_t \in \mathbb{R}$, where $t \in \mathbb{Z}_+$ indexes time. The internal state at time t of the system is parameterized by a vector $x_t \in \mathbb{R}^m$, and its relationship to the output is described by

$$\begin{aligned} x_{t+1} &= Ax_t + Bu_t \\ y_t &= Cx_t. \end{aligned}$$

Here C is a fixed matrix, while x_0, A, and B are unknown parameters.

Linear system identification is the task of learning these unknown parameters from input-output data—that is a sequence of inputs $u_1, \ldots, u_T$ and the observed sequence of outputs $y_1, \ldots, y_T$ [105, 50]. We pose this task as a sparse inverse problem. Each source is a small LTI system with 2-dimensional state—the measurement model gives the output of the small system on the given input. To be concrete, the parameter space Θ is all tuples of the form (x_0, r, α, B) where x_0 and B both lie in the ℓ_∞ unit ball in $\mathbf{R}^2$, r is in $[0,1]$, and α is in $[0,\pi]$. The LTI system that each source describes has

$$A = r\begin{bmatrix} \cos(\alpha) & -\sin(\alpha) \\ \sin(\alpha) & \cos(\alpha) \end{bmatrix}, \qquad C = \begin{bmatrix} 1 & 0 \end{bmatrix}.$$

The mapping ϕ from the parameters (x_0, r, α, B) to the output of the corresponding LTI system on input $u_1, \ldots, u_T$ is differentiable. In terms of the overall LTI system, adding the output of two weighted sources corresponds to concatenating the corresponding parameters. Again, a reasonable choice for the loss function is the squared error between the expected observations and the actual observations. In §3.3, we show that our algorithm matches the state of the art on two standard system identification datasets.

Matrix completion. The task of matrix completion is to estimate all entries of a large matrix given observations of a few entries. Clearly this task is impossible without prior information or assumptions about the matrix. If we believe that a low-rank matrix will approximate the truth well, a common heuristic is to minimize the squared error subject to a nuclear norm bound. For background in the theory and practice of matrix completion under this assumption see [7, 20]. We solve the following optimization problem:

$$\min_{\|A\|_* \leq \tau} \|M(A) - y\|^2.$$

Here M is the masking operator, that is, the linear operator that maps a matrix $A \in \mathbb{R}^{n \times m}$ to the vector containing its observed entries, and y is the vector of observed entries. We can rephrase this in our notation by letting $\Theta = \{(u, v) \in \mathbf{R}^n \times \mathbf{R}^m : \|u\|_2 = \|v\|_2 = 1\}$, $\phi((u, v)) = M(uv^T)$, and $\ell(\cdot) = \| \cdot - y\|^2$. In §3.3, we show that our algorithm achieves state of the art results on the Netflix Challenge, a standard benchmark in matrix completion.

Bayesian experimental design. In experimental design we seek to estimate a vector $x \in \mathbf{R}^d$ from measurements of the form

$$y_i = f(\theta_i)^T x + \epsilon_i.$$

Here $f : \Theta \to \mathbf{R}^d$ is a known differentiable feature function and ϵ_i are independent noise terms. We want to choose $\theta_i, \ldots, \theta_k$ to minimize our uncertainty about x — if each measurement requires a costly experiment, this corresponds to getting the most information from a fixed number of experiments. For background, see [91].

In general, this task in intractable. However, if we assume ϵ_i are independently distributed as standard normals and x comes from a standard normal prior we can analytically derive the posterior distribution of x given $y_1, \ldots, y_k$, as the full joint distribution of $x, y_1, \ldots, y_k$ is normal.

One notion of how much information $y_1, \ldots, y_k$ carry about x is the entropy of the posterior distribution of x given the measurements. We can then choose $\theta_1, \ldots, \theta_k$ to minimize the entropy of the posterior, which is equivalent to minimizing the (log) volume of an uncertainty ellipsoid. With this setup, the posterior entropy is (up to additive constants and a positive multiplicative factor) simply

$$-\log\det\left(I + \sum_i f(\theta_i) f(\theta_i)^T\right)^{-1}.$$

To put this in our framework, we can take $\phi(\theta) = f(\theta)f(\theta)^T$ and $\ell(M) = -\log\det(I + M)^{-1}$. We relax the requirement to choose exactly k measurement parameters and instead search for a non-negative sparse measure with bounded total mass, giving us an instance of (2.4).

Fitting mixture models to data. Given a parametric distribution $P(x|\theta)$ we consider the task of recovering the components of a mixture model from i.i.d. samples. For background see [69]. To be more precise, we are given data $\{x_1, \ldots, x_d\}$ sampled i.i.d. from a distribution of the form $P(x) = \int_{\theta\in\Theta} P(x|\theta)\pi(\theta)$. The task is to recover the mixing distribution π. If we assume π is sparse, we can phrase this as a sparse inverse problem. To do so, we choose $\phi(\theta) = (P(x_i|\theta))_{i=1}^d$. A common choice for ℓ is the (negative) log-likelihood of the data: i.e., $\ell(p) = -\sum_i \log p_i$. The obvious constraints here are $\int d\pi(\theta) \le 1, \pi \ge 0$.

Design of numerical quadrature rules. In many numerical computing applications we require fast procedures to approximate integration against a fixed measure. One way to do this is use a quadrature rule:

$$\int f(\theta)dp(\theta) \simeq \sum_{i=1}^{k} w_i f(x_i).$$

The quadrature rule, given by $w_i \in \mathbf{R}$ and $x_i \in \Theta$, is chosen so that the above approximation holds for functions f in a certain function class. The pairs (x_i, w_i) are known as quadrature nodes. In practice, we want quadrature rules with very few nodes to speed evaluation of the rule.

Often we don't have an a priori description of the function class from which f is chosen, but we might have a finite number of examples of functions in the class, $f_1, \ldots, f_d$, along with their integrals against p, $y_1, \ldots, y_d$. In other words, we know that

$$\int f_i(\theta)dp(\theta) = y_i.$$

A reasonable quadrature rule should approximate the integrals of the known f_i well.

We can phrase this task as a sparse inverse problem where each source is a single quadrature node. In our notation, $\phi(\theta) = (f_1(\theta), \ldots, f_d(\theta))$. Assuming each function f_i is differentiable, ϕ is differentiable. A common choice of ℓ for this application is simply the squared difference from y. For more discussion of the design of quadrature rules using the conditional gradient method, see [8, 70].

Neural spike sorting. In this example we consider the voltage v recorded by an extracellular electrode implanted in the vicinity of a population of neurons. Suppose that this population of neurons contains T types of neurons, and that when a neuron of type k fires at time $t \in \mathbb{R}$, an action potential of the form $\phi(t, k)$ is recorded. Here $\phi : [0, 1] \times \{1, \ldots, T\} \to \mathbb{R}^d$

is a vector of voltage samples. Note that ϕ is not differentiable in its last argument. The algorithms we discuss in this thesis can still be applied in this case, but any steps leveraging differentiability of ϕ must operate only on its first argument. If we denote the parameters of the ith neuron by $\theta_i = (t_i, k_i)$, then the total voltage $v \in \mathbb{R}^d$ can be modeled as a superposition of these action potentials:

$$v = \sum_{i=1}^{K} w_i \phi(\theta_i).$$

Here the weights $w_i > 0$ can encode the distance between the ith neuron and the electrode. The sparse inverse problem in this application is to recover the parameters $\theta_1, \ldots, \theta_K$ and weights $w_1, \ldots, w_K$ from the voltage signal v. For background see [37].

Designing radiation therapy. External radiation therapy is a common treatment for cancer in which several beams of radiation are fired at the patient to irradiate tumors. The collection of beam parameters (their intensities, positions, and angles) is called the treatment plan, and is chosen to minimize an objective function specified by an oncologist. The objective usually rewards giving large doses of radiation to tumors, and low dosages to surrounding healthy tissue and vital organs. Plans with few beams are desired as repositioning the emitter takes time—increasing the cost of the procedure and the likelihood that the patient moves enough to invalidate the plan.

A beam fired with intensity $w > 0$ and parameter θ delivers a radiation dosage $w\phi(\theta) \in \mathbb{R}^d$. Here the output is interpreted as the radiation delivered to each of d voxels in the body of a patient. The radiation dosage from beams with parameters $\theta_1, \ldots, \theta_K$ and intensities $w_1, \ldots, w_K$ add linearly, and the objective function is convex. For background see [57].

2.3 Relationship to atomic norm problems

Problems similar to (2.4) have been widely studied through the lens of atomic norms [21]. In this section we review the definition of an atomic norm and examine the intimate connection between (2.4) and a particular atomic norm problem. We discuss the advantages and disadvantages of each formulation.

The connection we make in this section is analogous to the simple observation that all quadratic norms are equivalent to the standard ℓ_2-norm after a linear transformation. That is, for every quadratic norm defined by

$$\|x\|_P^2 = x^T P x$$

for a positive-definite matrix P, we have that

$$\|x\|_P^2 = \inf_z \{\|z\|_2^2 \;:\; Az = x\}.$$

In this case A is a any square root of P^{-1} (including rectangular square roots where z is in a higher-dimensional space than x). As we will see, in the case of atomic norms and sparse linear inverse problems we have an analogous identity:

$$\|x\|_{\mathcal{A}} = \inf_{\mu}\{\|\mu\|_1 : \Phi\mu = x\}.$$

Just as P has many square roots in the quadratic norm case, there are many possible choices for Φ (and Θ) in the atomic norm case. In this and the following chapter, we argue that for atomic norms generated by an underlying sparse linear inverse problems,(2.4) is the most natural formulation.

An analogous atomic norm problem

The atomic norm $\|\cdot\|_{\mathcal{A}}$ corresponding to a suitable collection of atoms $\mathcal{A} \subset \mathbf{R}^d$ is the Minkowski functional of $\text{conv}(\mathcal{A})$, defined by

$$\|x\|_{\mathcal{A}} = \inf\{\tau \geq 0 : x \in \tau\text{conv}(\mathcal{A})\}.$$

Here $\text{conv}(\mathcal{A})$ is the convex hull of $\mathcal{A}$. The atomic norm, while not properly a norm without additional restrictions [21], is always a convex function. Much research has gone into the problem of estimating a vector $x \in \mathbf{R}^d$ that is believed to be a sum of a few elements of $\mathcal{A}$, using the atomic norm as a regularizer [21, 95].

It is tempting to approach (2.2) using the atomic norm generated by the set $\mathcal{A} = \{\pm\phi(\theta) : \theta \in \Theta\}$. Indeed, the noiseless signal x is a sum of a few (weighted) elements of $\mathcal{A}$. The atomic norm analogue to (2.4) is given by

$$\begin{array}{ll} \text{minimize} & \ell(x) \\ \text{subject to} & x \in \tau\text{conv}(\mathcal{A}). \end{array} \tag{2.5}$$

We'll show that the infinite-dimensional optimization problem (2.4) and the finite-dimensional atomic norm problem (2.5) are equivalent (in the sense of optimal objective value) under the (linear) change of variables

$$\mu \mapsto \Phi\mu. \tag{2.6}$$

Lemma 1 shows that the feasible set of (2.5) is exactly the image of the feasible set of (2.4) under the linear transformation Φ, which implies that the optimal objective values of (2.4) and (2.5) are the same, and that the solutions are linked by $x_\star = \Phi\mu_\star$.

Lemma 1. $\text{conv}(\mathcal{A}) = \{\Phi\mu : \|\mu\|_1 \leq 1\}$

Proof. It's clear that $\text{conv}(\mathcal{A}) \subset \{\Phi\mu : \|\mu\|_1 \leq 1\}$: the convex hull is smallest convex set containing $\{\pm\phi(\theta) : \theta \in \Theta\}$, which are the images of the measures $\pm\delta_\theta$ under Φ.

The other direction is slightly more difficult. Suppose $x = \int \phi(\theta)d\mu(\theta)$, with $\|\mu\|_1 \leq 1$ and $x \notin \text{conv}(\mathcal{A})$. Now we'd like to strongly separate $\{x\}$ from $\text{conv}(\mathcal{A})$ using Corollary 1.4.2

from [99]. This requires x to be outside of the closure of $\mathrm{conv}(\mathcal{A})$. Fortunately, $\mathrm{conv}(\mathcal{A})$ is already closed. To see this, note that $\mathcal{A}$ is compact as it is the union of two sets, each of which is compact as they are the image of the compact set Θ under the continuous function ϕ. Finally, as $\mathcal{A}$ is a compact set in a finite-dimensional space, Caratheodory's theorem implies that its convex hull is also compact [99, Theorem 17.2].

Then by [99, Corollary 1.4.2], there exists a hyperplane separating x and $\mathrm{conv}(\mathcal{A})$ strongly. Let v be the normal vector to the separating hyperplane. Then we have $\langle v, x\rangle > \sup_\theta \pm\langle v, \phi(\theta)\rangle = \sup_\theta |\langle v, \phi(\theta)\rangle|$. This is a contradiction as $\langle v, x\rangle = \langle v, \int \phi(\theta) d\mu(\theta)\rangle = \int \langle v, \phi(\theta)\rangle d\mu(\theta) \leq \sup_\theta |\langle v, \phi(\theta)\rangle| \|\mu\|_1 \leq \sup_\theta |\langle v, \phi(\theta)\rangle|$. □

Sparse Solutions

Lemma 1 suggests a simple method to prove that (2.4) always has a sparse solution: apply Caratheodory's theorem to the set $\{\Phi\mu : \|\mu\|_1 \leq 1\}$.

Lemma 2. (2.4) *has at least one sparse solution.*

Proof. Let $\mu_\star$ be a solution of (2.4). Then $\Phi\mu_\star \in \tau\mathrm{conv}(\{\pm\phi(\theta) : \theta \in \Theta\})$. Caratheodory's theorem for convex hulls then implies that $\Phi\mu_\star = \sum_{i=1}^{d+1} w_i \phi(\theta_i)$ with $\|w\|_1 \leq \tau$, $\theta_i \in \Theta$. Let $\mu = \sum_i w_i \delta_{\theta_i}$. Then μ is sparse and optimal for (2.4). □

Discussion

Much of the literature on sparse inverse problems focuses on problem (2.5), as opposed to the infinite-dimensional problem (2.4). This focus is due to the fact that (2.5) has algorithmic and theoretical advantages over (2.4). First and foremost, (2.5) is finite-dimensional, which means that standard convex optimization algorithms may apply. Additionally, the geometry of the atomic norm ball, $\mathrm{conv}\{\pm\phi(\theta) : \theta \in \Theta\}$, gives clean geometric insight into when the convex heuristic will work [21].

With that said, we hold that the infinite-dimensional formulation we study has distinct practical and theoretical advantages over the atomic norm problem (2.5), at least when one is interested in (2.3). A solution $x_\star$ to the atomic norm problem is an estimate of the vector $\Phi\mu_{\text{true}}$, while what we actually seek is an estimate of the signal parameters $\{(w_1, \theta_1), \ldots, (w_k, \theta_k)\}$. In many applications, it is this atomic decomposition μ that is of interest, and *not* the optimal point $x_\star$ of (2.5). Reconstructing an optimal $\mu_\star$ for problem (2.4) from $x_\star$ can be highly nontrivial; for many measurement models this is as hard as solving (2.4). As such, an algorithm that returns $x_\star$ is useless in the context of solving the original signal estimation task. For example, when designing radiation therapy, the measure $\mu_\star$ encodes the optimal beam plan directly, while the vector $x_\star = \Phi\mu_\star$ is simply the pattern of radiation that the optimal plan produces. Likewise, in superresolution microscopy, $\mu_\star$ encodes the locations and intensities of the localized fluorophores, while $x_\star$ is the *blurry* image they produce. For this reason, an algorithm that simply returns the vector $x_\star$, without the underlying atomic decomposition, is not always useful in practice.

Additionally, the measure-theoretic framework exposes the underlying parameter space, which in many applications comes with meaningful and useful structure. Furthermore, in many applications the measurement operator ϕ is known, but there may be no way to compute $\|x\|_{\mathcal{A}}$ — meaning that many standard convex optimization algorithms cannot be immediately applied to the atomic norm problem. Finally, naïve interpretation of the finite-dimensional optimization problem treats the parameter space as an unstructured set; keeping the structure of the parameter space in mind makes extensions such as ADCG that make local movements in parameter space natural and uniform across applications.

Chapter 3

Algorithms for Sparse Linear Inverse Problems

This chapter presents a variant of the classical conditional gradient method for sparse linear inverse problems first proposed in:

N. Boyd, G. Schiebinger, and B. Recht. "The Alternating Descent Conditional Gradient Method for Sparse Inverse Problems". In: *SIAM Journal on Optimization* 27.2 (2017), pp. 616–639.

Our algorithm combines nonconvex and convex optimization techniques: we propose global conditional gradient steps alternating with nonconvex local search exploiting the differentiable observation model. This hybridization gives the theoretical global optimality guarantees and stopping conditions of convex optimization along with the performance and modeling flexibility associated with nonconvex optimization. Our experiments demonstrate that our technique achieves state-of-the-art results in several applications.

3.1 Conditional gradient method

In this section we present our main algorithmic development. We begin with a review of the classical conditional gradient method (CGM) for finite-dimensional convex programs. We then apply the CGM to the sparse inverse problem (2.4). In particular, we augment this algorithm with a local search subroutine that significantly improves the practical performance.

The classical CGM solves the following optimization problem:

$$\underset{x \in \mathcal{C}}{\text{minimize}} \; f(x), \tag{3.1}$$

where $\mathcal{C}$ is a bounded convex set and f is a differentiable convex function.

The CGM proceeds by iteratively solving linearized versions of (3.1). At iteration k, we form the standard linear approximation to the function f at the current point x_k:

$$\hat{f}_k(s) = f(x_k) + f'(s - x_k; x_k).$$

Here $f'(s-x_k;x_k)$ is the directional derivative of the function f at x_k in the direction $s-x_k$. When f is differentiable, $f'(s-x_k;x_k)$ is more commonly written as $\langle \nabla f(x_k), s-x_k\rangle$; we use the directional derivative to make the extension to optimization on measures easier. As f is convex, this approximation is a global lower bound. We then minimize the linearization over the feasible set to get a potential solution s_k. As s_k minimizes a simple approximation of f that degrades with distance from x_k we take a convex combination of s_k and x_k as the next iterate. We summarize this method in Algorithm 1.

```
Algorithm 1 Conditional gradient method (CGM)
For $k = 1, \dots k_{\max}$
    1. Linearize: $\hat{f}_k(s) \leftarrow f(x_k) + f'(s-x_k;x_k)$.
    2. Minimize: $s_k \ni \arg\min_{s\in\mathcal{C}} \hat{f}_k(s)$.
    3. Tentative update: $\tilde{x}_{k+1} \leftarrow \frac{k}{k+2}x_k + \frac{2}{k+2}s_k$.
    4. Final update: Choose $x_{k+1}$ such that $f(x_{k+1}) \le f(\tilde{x}_{k+1})$.
```

It is important to note that minimizing $\hat{f}_k(s)$ over the feasible set $\mathcal{C}$ in step 2 may be quite difficult and requires an application-specific subroutine.

One of the more remarkable features of the CGM is step 4. While the algorithm converges using only the tentative update in step 3, all of the convergence guarantees of the algorithm are preserved if one replaces $\tilde{x}_{k+1}$ with *any* feasible x_{k+1} that achieves a smaller value of the objective. There are thus many possible choices for the final update in step 4, and the empirical behavior of the algorithm can be quite different for different choices. One common modification is to do a line-search:

$$x_{k+1} = \underset{x\in\operatorname{conv}(x_k,s_k)}{\arg\min}\ f(x).$$

We use conv to denote the convex hull—in this last example, a line segment. Another variant, the *fully-corrective* conditional gradient method, chooses

$$x_{k+1} = \underset{x\in\operatorname{conv}(x_k,s_1,\dots,s_k)}{\arg\min}\ f(x).$$

In the next section, we propose a natural choice for this step in the case of measures that uses local search to speed-up the convergence of the CGM.

One appealing aspect of the CGM is that it is very simple to compute a lower bound on the optimal value $f_\star$ as the algorithm runs. As $\hat{f}_k$ lower-bounds f, we have

$$f(s) \ge \hat{f}_k = f(x_k) + f'(s-x_k;x_k) = \hat{f}_k(s)$$

for any $s \in \mathcal{C}$. Minimizing both sides over s gives us the elementary bound

$$f_\star \geq \hat{f}_k(s_k).$$

The right hand side of this inequality is readily computed after step (2). One can prove that the bound on suboptimality derived from this inequality decreases to zero [60], which makes it a very useful termination condition.

CGM for sparse linear inverse problems

In this section we apply the classical CGM to the sparse inverse problem (2.4). We give two versions—first a direct translation of the fully corrective variant and then our improved algorithm that leverages local search on Θ. To make it clear that we operate over the space of measures on Θ we change notation and denote the iterate by μ_k instead of x_k. The most obvious challenge is that we cannot easily represent a general measure on a computer. However, we will see that the steps of CGM can in fact be carried out on a computer in this context. In fact, each iterate is a sparse measure μ_k supported on N_k points:

$$\mu_k = \sum_{i=1}^{N_k} w_i^{(k)} \delta_{\theta_i^{(k)}}.$$

As such, we will represent μ_k by the pair of vectors: $w_k \in \mathbf{R}^{N_k}$ and $\vec{\theta}_k \in \Theta^{N_k}$. Indeed, the algorithms we present can be thought of as operating on this representation directly.

Before we describe the algorithm in detail, we first explain how to linearize the objective function and minimize the linearization. In the space of measures, linearization is most easily understood in terms of the (one-sided) directional derivative.

In our formulation (2.4), $f(\mu) = \ell(\Phi\mu)$. If we define the *output* as $o_k = \Phi\mu_k$, we can compute the directional derivative of our particular choice of f at μ_k in the direction of the measure s as

$$f'(s; \mu_k) = \lim_{t \downarrow 0} \frac{\ell(\Phi(\mu_k + ts)) - \ell(\Phi\mu_k)}{t} = \lim_{t \downarrow 0} \frac{\ell(o_k + t\Phi s) - \ell(o_k)}{t} = \ell'(\Phi s; o_k) = \langle \nabla \ell(o_k), \Phi s \rangle . \tag{3.2}$$

Here, the inner product on the right hand side of the equation is the standard inner product in $\mathbb{R}^d$.

The second step of the CGM minimizes the linearized objective over the constraint set. In other words, we minimize $\langle \nabla \ell(o_k), \Phi s \rangle$ over a candidate measure s with total variation bounded by τ. Interchanging the integral (in Φ) with the inner product, and defining $F(\theta) := \langle \nabla \ell(o_k), \phi(\theta) \rangle$, we need to solve the optimization problem:

$$\underset{|s|(\Theta) \leq \tau}{\text{minimize}} \int F(\theta) ds(\theta). \tag{3.3}$$

An optimal solution of (3.3) is the point-mass $-\tau \text{sgn}(F(\theta_\star))\delta_{\theta_\star}$, where $\theta_\star \in \arg\max |F(\theta)|$. This is clear, as $\int F(\theta)ds(\theta)$ is bounded by $-\sup_\theta |F(\theta)| \|s\|_1$. This means that at each step of the CGM we need only add a single point to the support of our approximate solution μ_k.

We now describe the fully-corrective variant of the CGM for sparse inverse problems (Algorithm 2). The state of the algorithm at iteration k is an atomic measure μ_k supported on a finite set $\vec{\theta}_k$ with weights w_k. The algorithm alternates between selecting a source to add to the support and tuning the weights to lower the current cost. Step 4a is a finite-dimensional convex optimization problem that we can solve with an off-the-shelf algorithm. The solution to the finite-dimensional optimization problem may set some of w_{k+1} to zero, in which case the prune subroutine removes the corresponding entries from w_{k+1} and $\vec{\theta}_{k+1}$.

Algorithm 2 Conditional gradient method for measures (CGM-M)

```
For $k = 1 : k_{\max}$
    1. Compute gradient of loss:    $g_k = \nabla \ell(\Phi \mu_k)$.
    2. Compute next source:         $\theta_k \in \arg\max_{\theta \in \Theta} |\langle g_k, \phi(\theta) \rangle|$.
    3. Update support:              $\vec{\theta}_{k+1} \leftarrow [\vec{\theta}_k, \theta_k]$.
    4. Compute weights:             $w_{k+1} \leftarrow \arg\min_{\|w_{k+1}\|_1 \leq \tau} \ell(\Phi \mu_{k+1})$.
    5. Prune support:               $(w_{k+1}, \vec{\theta}_{k+1}) \leftarrow \text{prune}(w_{k+1}, \vec{\theta}_{k+1})$.
```

We stress here that the objective in step 2 is *nonlinear* (and nonconvex) in the parameter θ, but *linear* when considered as a functional of the measure s_k.

While we can simply run for a fixed number of iterations, we may stop early using the standard CGM bound. With a tolerance parameter $\epsilon > 0$, we terminate when the conditional gradient bound assures us that we are at most ϵ-suboptimal. In particular, we terminate when

$$\tau |\langle \phi(\theta_k), g_k \rangle| + \langle \Phi \mu_k, g_k \rangle < \epsilon. \tag{3.4}$$

Unfortunately, CGM-M does not perform well in practice. Not only does it converge very slowly, but the solution it finds is often supported on an undesirably large set. As illustrated in Figure 3.1, the performance of CGM-M is limited by the fact that it can only change the support of the measure by adding and removing points; it cannot smoothly move S_k within Θ. Figure 3.1 shows CGM-M applied to an image of two closely separated sources. The first source θ_1 is placed in a central position overlapping both true sources. In subsequent iterations sources are placed too far to the right and left, away from the true sources. To move the support of the candidate measure requires CGM-M to repeatedly add and remove sources; it is clear that the ability to move the support smoothly within the parameter space would resolve this issue immediately.

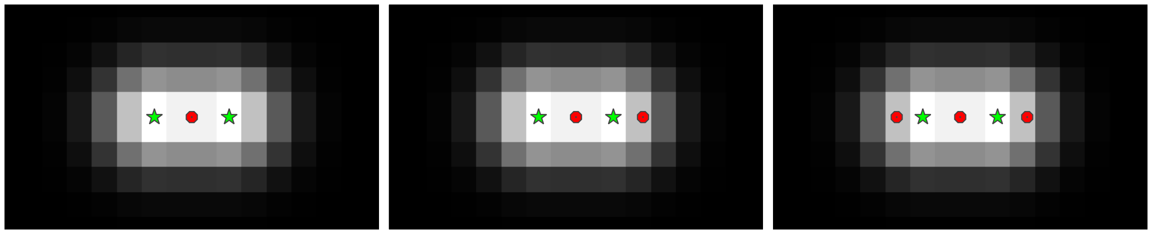


Figure 3.1: The three plots above show the first three iterates of the fully corrective CGM in a simulated superresolution imaging problem with two point sources of light. The locations of the true point sources are indicated by green stars, and the greyscale background shows the pixelated image. The elements of S_k for $k = 1, 2, 3$ are displayed by red dots.

In practice, we can speed up convergence and find significantly sparser solutions by allowing the support to move continuously within Θ. The following algorithm, which we call the alternating descent conditional gradient method (ADCG), exploits the differentiability of ϕ to locally improve the support at each iteration.

Algorithm 3 Alternating descent conditional gradient method (ADCG)

```
For $k = 1 : k_{\max}$
    1. Compute gradient of loss:        $g_k = \nabla \ell(\Phi \mu_k)$.
    2. Compute next source:             $\theta_k \in \arg\max_{\theta \in \Theta} |\langle \phi(\theta), g_k \rangle|$.
    3. Update support:                  $\vec{\theta}_{k+1} \leftarrow [\vec{\theta}_k, \theta_k]$.
    4. Coordinate descent on nonconvex objective:
       Repeat:
        a) Compute weights:             $w_{k+1} \leftarrow \arg\min_{\|w_{k+1}\|_1 \leq \tau} \ell\left(\Phi \mu_{k+1}\right)$.
        b) Prune support:               $(w_{k+1}, \vec{\theta}_{k+1}) \leftarrow \text{prune}(w_{k+1}, \vec{\theta}_{k+1})$.
        c) Locally improve support:     $\vec{\theta}_{k+1} = \textbf{local\_descent}(w_{k+1}, \vec{\theta}_{k+1})$.
```

Here **local_descent** is a subroutine that takes a measure μ with atomic representation $w, \vec{\theta}$ and attempts to use gradient information to reduce the function

$$(\theta_1, \ldots, \theta_m) \mapsto \ell\left(\sum_{i=1}^{m} w_i \phi(\theta_i)\right),$$

holding the weights fixed.

When the number of sources is held fixed, the optimization problem

$$\begin{array}{ll} \text{minimize} & \ell\left(\sum_{i=1}^{m} w_i \phi(\theta_i)\right) \\ \text{subject to} & \vec{\theta} \in \Theta^m \\ & \|w\|_1 \leq \tau \end{array} \tag{3.5}$$

is nonconvex. Step 4 is then block coordinate descent over w and $\vec{\theta}$. The algorithm as a whole can be interpreted as alternating between performing descent on the convex (but infinite-dimensional) problem (2.4) in step 2 and descent over the finite-dimensional (but nonconvex) problem (3.5) in step 4. The bound (3.4) remains valid and can be used as a termination condition. Note that while we use block coordinate descent for simplicity, other algorithms that perform descent over w and $\vec{\theta}$ simultaneously would also work well.

As we have previously discussed, this nonconvex local search does not change the convergence guarantees of the CGM whatsoever. We will show in §3.4 that this is an immediate consequence of the existing theory on the CGM. However, as we will show in §3.3, the inclusion of local search dramatically improves the performance of the CGM.

Interface and implementation

Roughly speaking, running ADCG on a concrete instance of (2.4) requires subroutines for two operations. We need algorithms to compute:

(a) $\phi(\theta)$ and $\frac{d}{d\theta}\phi(\theta)$ for $\theta \in \Theta$.

(b) $\arg\max_{\theta \in \Theta} |\langle \phi(\theta), v \rangle|$ for arbitrary vectors $v \in \mathbf{R}^d$.

Computing (a) is usually straightforward in applications with differentiable measurement models. Computing (b) is not easy in general. However, there are many applications of interest where (b) is tractable. For example, if the parameter space Θ is low-dimensional, then the ability to compute (a) is sufficient to approximately compute (b): we can simply grid the parameter space and begin local search using the gradient of the function $\theta \mapsto \langle \phi(\theta), v \rangle$. Note that because of the local improvement step, ADCG works well even without exact minimization of (b). We prove this fact about inexact minimization in §3.4.

If the parameter space is high-dimensional, however, the feasibility of computing (b) will depend on the specific application. One example of particular interest that has been studied in the context of the CGM is matrix completion [61, 95, 51, 124]. In this case, the (b) step reduces to computing the leading singular vectors of a sparse matrix. We will show that adding local improvement to the CGM accelerates its convergence on matrix completion in the experiments.

We also note that in the special case of linear system identification, Θ is 6 dimensional, which is just large enough such that gridding is not feasible. In this case, we show that we can reduce the 6-dimensional optimization problem to a 2-dimensional problem and then again resort to gridding. We expect that in many cases of interest, such specialized solvers can be applied to solve the selection problem (b).

3.2 Related work

There has recently been a renewed interest in the conditional gradient method as a general purpose solver for constrained inverse problems [60, 51]. These methods are simpler to implement than the projected or proximal gradient methods which require solving a quadratic rather than linear optimization over the constraint set.

The idea of augmenting the classic conditional gradient method with improvement steps is not unique to our work. Indeed, it is well known that any modification of the iterate that decreases the objective function will not hurt theoretical convergence rates [60]. Moreover, Rao *et al* [95] have proposed a version of the conditional gradient method, called CoGENT, for atomic norm problems that takes advantage of many common structures that arise in inverse problems. The reduction described in our theoretical analysis makes it clear that our algorithm can be seen as an instance of CoGENT specialized to the case of measures and differentiable measurement models.

The most similar proposals to ADCG come from the special case of matrix completion or nuclear-norm regularized problems. Several papers [124, 71, 51, 61] have proposed algorithms based on combinations of rank-one updates and local nonconvex optimization inspired by the well-known heuristic of [17]. While our proposal is significantly more general, ADCG essentially recovers these algorithms in the special case of nuclear-norm problems.

We note that in the context of inverse problems, there are a variety of algorithms proposed to solve the general infinite-dimensional problem (2.4). Tang *et al* [113] prove that this problem can be approximately solved by gridding the parameter space and solving the resulting finite dimensional problem. However, these gridding approaches are not tractable for problems with parameter spaces of even relatively modest dimension. Moreover, even when gridding is tractable, the solutions obtained are often supported on very large sets and heuristic post-processing is required to achieve reasonable performance in practice [113]. In spite of these limitations, gridding is the state of the art in many application areas including computational neuroscience [37], superresolution fluorescence microscopy [125], radar [10, 55], remote sensing [40], compressive sensing [9, 28, 33], and polynomial interpolation [96].

There have also been a handful of papers that attempt to tackle the infinite-dimensional problem without gridding. For the special case where $\ell(\cdot) = \|\cdot\|_2^2$, Bredies and Pikkarainen [16] propose an algorithm to solve the Tikhonov-regularized version of problem (2.4) that is very similar to Algorithm 3. They propose performing a conditional gradient step to update the support of the measure, followed by soft-thresholding to update the weights. Finally, with the weights of the measure fixed they perform discretized gradient flow over the locations of the point-masses. However, they do not solve the finite-dimensional convex problem at every iteration, which means there is no guarantee that their algorithm has bounded memory requirements. Much more seriously, for the same reason, they are limited to one pass of gradient descent in the nonconvex phase of the algorithm. In §3.3 we show that this limitation has serious performance implications in practice.

3.3 Numerical results

In this section we apply ADCG to three of the examples in §2.2: superresolution fluorescence microscopy, matrix completion, and system identification. We have made a simple implementation of ADCG publicly available on github:

https://github.com/nboyd/SparseInverseProblems.jl.

This allows the interested reader to follow along with these examples, and, hopefully, to apply ADCG to other instances of (2.4).

For each example we briefly describe how we implement the required subroutines for ADCG, though again the interested reader may want to consult our code for the full picture. We then describe how ADCG compares to prior art. Finally, we show how ADCG improves on the standard fully-corrective conditional gradient method for measures (CGM-M) and a variant of the gradient flow algorithm (GF) proposed in [16]. While the gradient flow algorithm proposed in [16] does not solve the finite-dimensional convex problem at each step, our version of GF does. We feel that this is a fair comparison: intuitively, fully solving the convex problem can only improve the performance of the GF algorithm. All three experiments require a subroutine to solve the finite-dimensional convex optimization problem over the weights. For this we use a simple implementation of a primal-dual interior point method, which we include in our code package.

For each experiment we select the parameter τ by inspection. For matrix completion and linear system ID this means using a validation set. For single molecule imaging each image requires a different value of τ. For this problem, we run ADCG with a large value of τ and stop when the decrease in the objective function gained by the addition of a source falls below a threshold. This heuristic can be viewed as post-hoc selection of τ and the stopping tolerance ϵ, or as a stagewise algorithm [116].

The experiments are run on a standard c4.8xlarge EC2 instance. Our naive implementations are meant to demonstrate that ADCG is easy to implement in practice and finds high-quality solutions to (2.4). For this reason we do not include detailed timing information.

Superresolution fluorescence microscopy

We analyze data from the Single Molecule Localization Microscopy (SMLM) challenge [102, 49]. Fluorescence microscopy is an imaging technique used in the biological sciences to study subcellular structures in vivo. The task is to recover the 2D positions of a collection of fluorescent proteins from images taken through an optical microscope.

Here we compare the performance of ADCG to the gridding approach of Tang *et al* [113], two algorithms from the microscopy community (quickPALM and center of Gaussians), and also CGM and the gradient flow (GF) algorithm proposed by [16]. The gridding approach approximately solves the continuous optimization problem (2.4) by discretizing the space Θ into a finite grid of candidate point source locations and running an ℓ_1-regularized regression. In practice there is typically a small cluster of nonzero weights in the neighborhood of each true point source. With a fine grid, each of these clusters contains many nonzero weights, yielding many false positives.

To remove these false positives, Tang *et al* [113] propose a heuristic post-processing step that involves taking the center of mass of each cluster. This post-processing step is hard to understand theoretically, and does not perform well with a high-density of fluorophores.

Implementation details

For this application, the minimization required in step 2 of ADCG is not difficult: the parameter space is two-dimensional. Coarse gridding followed by a local optimization method works well in theory and practice.

For **local_descent** we use a standard constrained gradient method provided by the NLopt library [63].

Evaluation

We measure localization accuracy by computing the F_1 score, the harmonic mean of precision and recall, at varying radii. Computing the precision and recall involves first matching estimated point sources to true point sources—a difficult task. Fortunately, the SMLM challenge website [49] provides a stand-alone application that we use to compute the F_1 score.

We use a dataset of 12000 images that overlay to form simulated microtubules (see Figure 3.2) available online at the SMLM challenge website [49]. There are 81049 point sources in total, roughly evenly distributed across the images. Figure 3.2a shows a typical image. Each image covers an area 6400nm across, meaning each pixel is roughly 100nm by 100nm.

Figure 3.3 compares the performance of ADCG, gridding, quickPALM, and center of Gaussians (CoG) on this dataset. We match the performance of the gridding algorithm from [113], and significantly beat both quickPALM and CoG. Our algorithm analyses all images in well under an hour—significantly faster than the gridding approach of [113]. Note that the gridding algorithm of [113] does not work without a post-processing step.

Matrix completion

As described in §2.2, matrix completion is the task of estimating an approximately low rank matrix from some of its entries. We test our proposed algorithm on the Netflix Prize dataset,

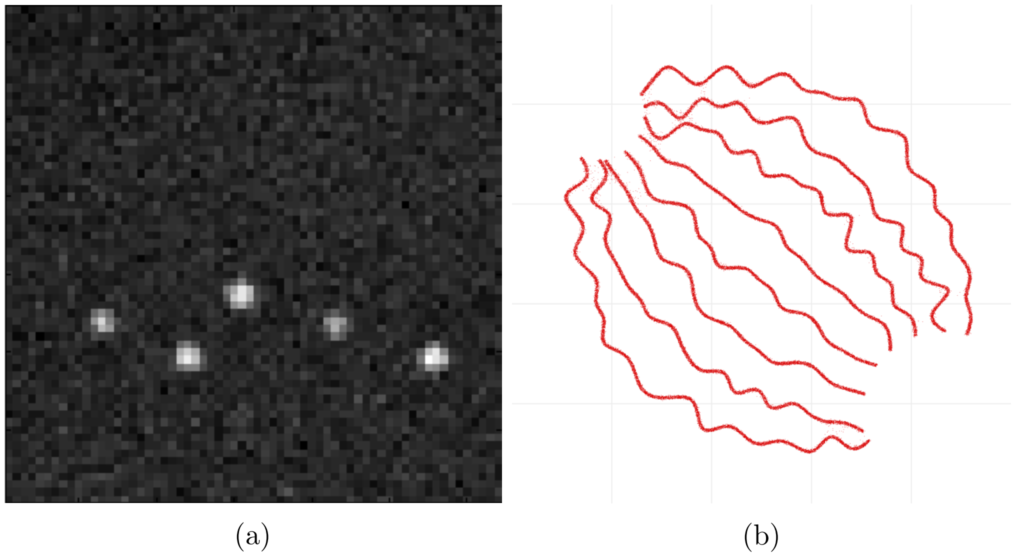


Figure 3.2: The long sequence dataset contains 12000 images similar to (a). The recovered locations for all the images are displayed in (b).

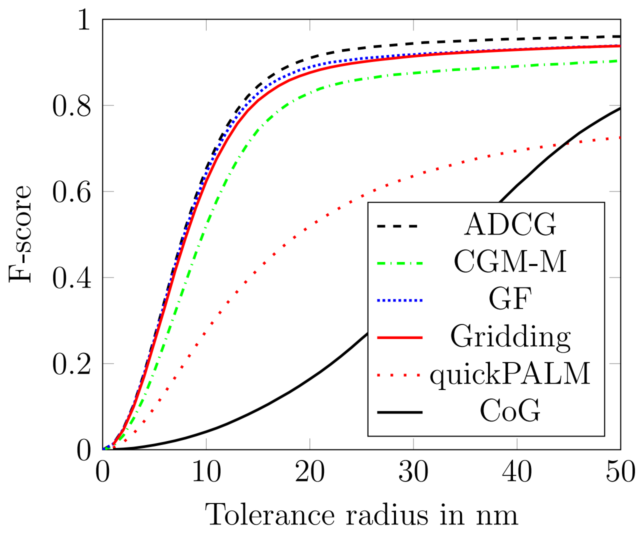


Figure 3.3: Performance on bundled tubes: long sequence. F-scores at various radii for 6 algorithms. For reference, each image is 6400nm across, meaning each pixel has a width of 100nm. ADCG outperforms all competing methods on this dataset.

a standard benchmark for matrix completion algorithms.

Implementation details

Although the parameter space for this example is high-dimensional we can still compute the steepest descent step over the space of measures. We need to minimize the following over a, b with $\|a\|_2 = \|b\|_2 = 1$:

$$\langle \phi(a,b), \nu \rangle = \langle M(ab^T), \nu \rangle = \langle ab^T, M^*(\nu) \rangle = a^T M^*(\nu) b.$$

In other words, we need to find the unit norm, rank one matrix with highest inner product with the matrix $M^*\nu$. The solution to this problem is given by the top singular vectors of $M^*\nu$. Computing the top singular vectors using a Lanczos method is relatively easy as the matrix $M^*\nu$ is extremely sparse.

Our implementation of **local_descent** takes a single step of gradient descent (on the sphere) with line-search.

Evaluation

Our algorithm matches the state of the art for nuclear norm based approaches on the Netflix Prize dataset. Briefly, the task here is to predict the ratings 480,189 Netflix users give to a subset of 17,770 movies. One approach has been to phrase this as a matrix completion problem. That is, to try to complete the 480,189 by 17,770 matrix of ratings from the observed entries. Following [97] we subtract the mean training score from all movies and truncate the predictions of our model to lie between 1 and 5.

Figure 3.4 shows root-mean-square error (RMSE) of our algorithm and other variants of the CGM on the Netflix probe set. Again, ADCG outperforms all other CGM variants. Our algorithm takes over 7 hours to achieve the best RMSE—this could be improved with a more sophisticated implementation, or parallelization.

Comparison to prior approaches

Many researchers have proposed solving matrix completion problems or general semi-definite programs using CGM-like algorithms; see [124, 71, 51, 61]. While ADCG applied to the matrix completion problem is distinct (to the best of our knowledge) from existing algorithms, it combines existing ideas. For instance, the idea of using the conditional gradient algorithm to solve the constrained formulation is very well known [61]. The idea of using local search on a low-rank factorization goes back at least to [17], and is used in many recent algorithms [124, 71].

In terms of performance, our implementation is relatively slow but gives very good validation error.

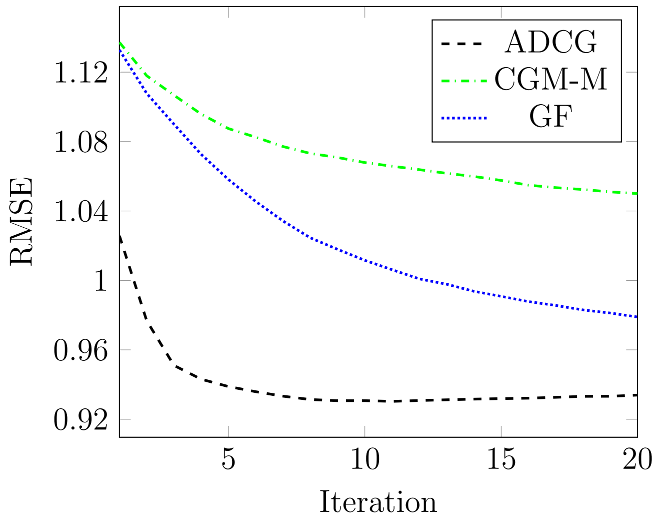


Figure 3.4: Test RMSE on Netflix challenge dataset. ADCG significantly outperforms CGM-M.

System identification

In this section we apply our algorithms to identifying two single-input single-output systems from the DaISy collection [11]: the flexible robot arm dataset (ID 96.009) and the hairdryer dataset (ID 96.006).

Implementation details

While the parameter space is 6-dimensional, which effectively precludes gridding, we can efficiently solve the minimization problem in step (2) of the ADCG. To do this, we grid only over r and α: the output is linear in the remaining parameters (B and x_0) allowing us to analytically solve for the optimal B and x_0 as a function of r and α.

For **local_descent** we again use a standard box-constrained gradient method provided by the NLopt library [63].

Evaluation

Both datasets were generated by driving the system with a specific input and recording the output. The total number of samples is 1000 in both cases. Following [105] we identify the system using the first 300 time points and we evaluate performance by running the identified system forward for the remaining time points and compare our predictions to the ground truth.

We evaluate our predictions y_{pred} using the score defined in [50]. The score is given by

$$\text{score} = 100\left(1 - \frac{\|y_{\text{pred}} - y\|_2}{\|y_{\text{mean}} - y\|_2}\right), \tag{3.6}$$

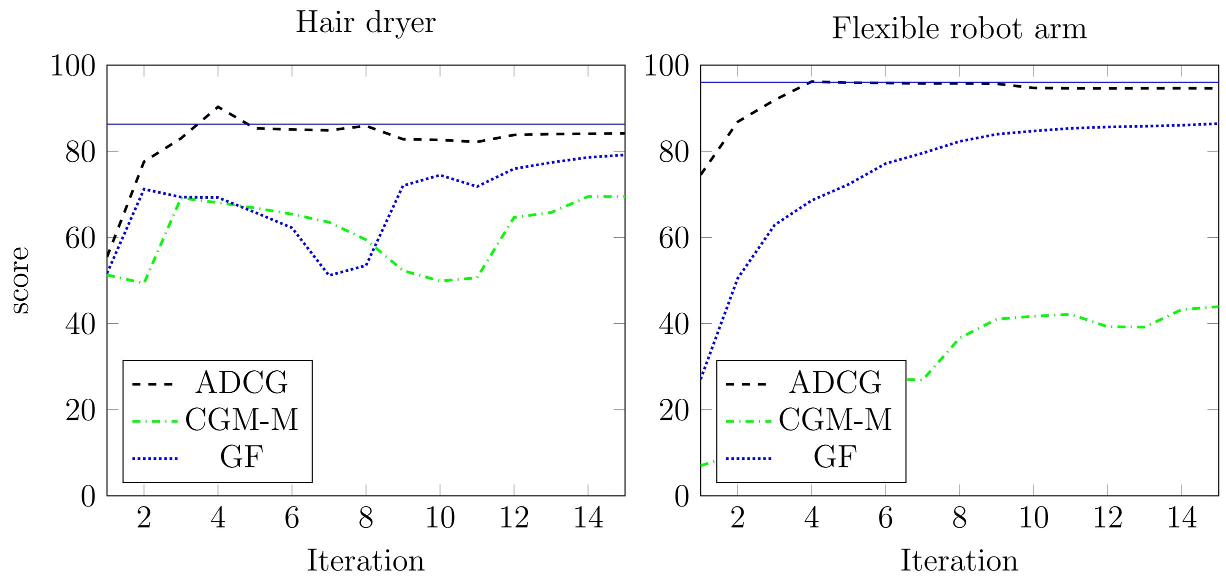


Figure 3.5: Performance on DaISy datasets. ADCG outperforms other CGM variants and matches the nuclear-norm based technique of [105].

where y_{mean} is the mean of the test set y.

Figure 3.5 shows the score versus the number of sources as we run our algorithm. For reference we display with horizontal lines the results of [50]. ADCG matches the performance of [50] and exceeds that of all other CGM variants. Our simple implementation takes about an hour, which compares very poorly with the spectral methods in [50] which complete in under a minute.

3.4 Theoretical guarantees

In this section we present a two simple theoretical results. The first guarantees that we can run our algorithm with bounded memory — though the bound is of limited use in practice, where our algorithm (typically) terminates in far fewer than $d + 1$ iterations. The second result guarantees that the algorithm converges to an optimal point and bounds the worst-case rate of convergence. Again, though, this rate of convergence is much slower than the rate observed in practice. We emphasize that the main contribution of this chapter is an algorithm that is effective in practice, and both of these guarantees are immediate consequences of existing theory.

Bounded memory

As the CGM for measures adds one point to the support of the iterate per iteration, we know that the cardinality of the support of μ_k is bounded by k. For large k, then, μ_k

could have large support. The following theorem guarantees that we can run our algorithm with bounded memory and in fact we need only store at most $d+1$ points, where d is the dimension of the measurements.

Theorem 3. *ADCG may be implemented to generate iterates with cardinality of support uniformly bounded by $d+1$.*

Proof. Lemma (4) allows us to conclude that the fully-corrective step ensures that the support of the measure remains bounded by $d+1$ for all iterations. $\square$

Lemma 4. *The finite-dimensional problem*

$$\underset{\|w\|_1 \le \tau}{\text{minimize}} \quad \ell\Big(\sum_i w_i \phi(\theta_i) - y\Big) \tag{3.7}$$

has an optimal solution $w_\star$ with at most $d+1$ nonzeros.

Proof. Let $u_\star$ be any optimal solution to (3.7). As $u_\star$ is feasible, we have that

$$v = \sum_i u_{\star i}\phi(\theta_i) \in \tau \mathrm{conv}(\{\pm\phi(\theta_i) : i = 1, \dots, m\}).$$

In other words, $\frac{v}{\tau}$ lies in the convex hull of a set in $\mathbf{R}^d$. Caratheodory's theorem immediately tells us that $\frac{v}{\tau}$ can be represented as a convex combination of at most $d+1$ points from $\{\pm\phi(\theta_i) : i = 1, \dots, m\}$. That is, there exists a $w_\star$ with at most $d+1$ nonzeros such that

$$\sum_{i=1}^m w_{\star i}\phi(\theta_i) = v.$$

This implies that $w_\star$ is also optimal for (3.7). $\square$

Note that in order to find $w_\star$, we need to either use a simplex-type algorithm to solve (3.7) or explore the optimal set using the random ray-shooting procedure as described in [108].

Convergence analysis

We now analyze the worst-case convergence rate for ADCG applied to (2.4). Note that the discussion in the last section on atomic norms (§2.3) implies that we can bound the convergence of ADCG by the convergence of the standard CGM applied to (2.5). With that said, standard proofs of the convergence of the conditional gradient method (known since the 1960's)[35, 30, 60] apply to any optimization problem of the following form in any Banach space:

$$\underset{x \in \mathcal{S}}{\text{minimize}} \;\; f(x). \tag{3.8}$$

Here $\mathcal{S}$ is a bounded convex set, and f is a convex function. The convergence result depends on a curvature parameter of the function f over the set $\mathcal{S}$, C_f. C_f is a constant such that the following inequality is satisfied for all $x, s \in \mathcal{S}$ and $\eta \in (0, 1)$:

$$f(x + \eta(s - x)) \leq f(x) + \eta f'(s - x; x) + \frac{C_f}{2}\eta^2.$$

Intuitively, C_f controls the quality of the linear approximation to f made in each iteration of the CGM. The standard convergence result is stated below:

Theorem 5. *Let C_f be the curvature parameter of the convex function f on the bounded, convex set $\mathcal{C}$. Let $x_1, x_2, \ldots$ be the iterates of the standard CGM applied to (3.8). Let $f_\star$ be the optimal value of (3.8). Then we have*

$$f(x_k) - f_\star \leq \frac{C_f}{k+2}.$$

In our setting we can simply take $\mathcal{S} = \{\mu : \|\mu\|_1 \leq \tau\}$ and $f(\mu) = \ell(\Phi\mu - y)$. The affine invariance of the conditional gradient method implies that the curvature of f over $\mathcal{S}$ is equal to the curvature of the function $g(x) = \ell(x - y)$ over the set $\mathcal{A} = \operatorname{conv}(\{\pm\phi(\theta) : \theta \in \Theta)\})$. As noted in [60], Theorem 5 applies to any algorithm that reduces the objective value at each iteration at least as much as the standard CGM. As ADCG falls into this category, it also converges at the specified rate.

The theorem applies even when the linear minimization step is performed approximately [60]. That is, we allow θ_k to be chosen such that

$$|\langle \phi(\theta_k), g_k \rangle| \leq \max_{\theta \in \Theta} |\langle \phi(\theta), g_k \rangle| + \frac{\zeta}{k+2} \tag{3.9}$$

for some $\zeta \geq 0$. When inequality (3.9) holds, we say that the linear minimization problem in iteration k is solved to precision ζ. When the linear minimization problem is solved to precision ζ, the convergence rate above applies when multiplied by the factor $(1 + \zeta)$.

3.5 Conclusions and future work

As demonstrated in the numerical experiments of §3.3, ADCG achieves state of the art performance in superresolution fluorescence microscopy, matrix completion, and system identification, without the need for heuristic post-processing steps. The addition of the nonconvex local search step significantly improves performance relative to the standard conditional gradient algorithm in all of the applications investigated. In some sense, we can understand ADCG as a method to rigorously control local search. One could just start with a model expansion (2.1) and perform nonconvex local search. However, this fares far worse than ADCG in practice and has no theoretical guarantees. The ADCG framework provides a clean way to generate a globally convergent algorithm that is practically efficient. Understanding this coupling between local search heuristics and convex optimization leads our brief discussion of future work.

Tighten convergence analysis for ADCG. The conditional gradient method is a robust technique, and adding our auxiliary local search step does not worsen its theoretical convergence rate. However, in practice, the difference between the ordinary conditional gradient method, the fully corrective variants, and ADCG are striking. In many of our experiments, ADCG outperforms the other variants by appreciable margins. Yet, all of these algorithms share the same upper bound on their convergence rate. A very interesting direction of future work would be to investigate if the bounds for ADCG can be tightened at all to be more predictive of practical performance. There may be connections between our algorithm and other alternating minimization techniques popular in matrix completion [64, 62], sparse coding [1, 3], and phase retrieval [83], and perhaps the techniques from this area could be applied to our setting of sparse inverse problems.

Connections to clustering algorithms. Another possible connection that could be worth exploring is the connection between the CGM and clustering algorithms like k-means. Theoretical bounds have been devised for initialization schemes for clustering algorithms that resemble the first step of CGM [4, 86]. In these methods, k-means is initialized by randomly seeking the points that are farthest from the current centers. This is akin to the first step of CGM which seeks the model parameters that best describe the residual error. Once a good seeding is acquired, the standard Lloyd iteration for k-means can be shown to converge to the global optimal solution [86]. It is possible that these analyses could be generalized to analyze our version of CGM or inspire new variants of the CGM.

Connections to cutting plane methods and semi-infinite programs. The standard Lagrangian dual of (2.4) is a semi-infinite program (SIP), namely an optimization problem with a finite dimensional decision variable but an infinite collection of constraints [56, 106]. One of the most popular algorithmic techniques for SIP is the cutting plane method, and these methods qualitatively act very much like the CGM. Exploring this connection in detail could generate variants of cutting plane methods suited for continuous constraint spaces. Such algorithms could be valuable tools for solving semi-infinite programs that arise in contexts disjoint from sparse inverse problems.

Other applications. We believe that our techniques are broadly applicable to other sparse inverse problems, and hope that future work will explore the usefulness of ADCG in areas unexplored in this thesis. To facilitate the application of ADCG to more problems, such as those described in §2.2, we have made our code publicly available on GitHub. As described in §3.1, implementing ADCG for a new application essentially requires only two user-specified subroutines: one routine that evaluates the observation model and its derivatives at a specified set of weights and model parameters, and one that approximately solves the linear minimization in step 2 of ADCG. We aim to investigate several additional applications in the near future to test the breadth of the efficacy of ADCG.

Chapter 4

Saturating Splines and Feature Selection

This chapter is based on the paper:

N. Boyd, T. Hastie, S. Boyd, B. Recht, and M. Jordan. "Saturating Splines and Feature Selection". In: *Journal of Machine Learning Research* 17.178 (2017).

In this chapter we explore, in detail, a specific sparse linear inverse problem that arises in statistics: fitting adaptive splines. We show that ADCG is a natural fit for this application. We show a slight modification of the adaptive spline problem — namely the addition of saturation constraints — is especially effective when fitting generalized additive models, and can be handled by a slight modification of ADCG. As a note, in this application the measure μ has a natural interpretation as the derivative of a non-differentiable function.

4.1 Introduction

Splines—piecewise polynomials with continuity constraints—are widely used to fit data [53, §5.1]. One issue with piecewise polynomials is that they behave erratically beyond their boundary knot points, and (typically) grow without bound outside of that range [53, §5.2]. This instability makes extrapolation dangerous; practitioners must take care to avoid querying spline models near or outside of the range of the training data.

Smoothing spline algorithms [29, 119, 46] ameliorate this problem by fitting *natural splines*, which reduce to a lower-degree polynomial beyond the boundary knots. The most commonly used varieties of smoothing splines are cubic smoothing splines (degree-three splines that reduce to linear outside the boundary knots) and linear smoothing splines, which extend as constant. The saturating splines we propose are closely related to linear smoothing splines.

Smoothing splines use an ℓ_2 or quadratic notion of complexity, and hence fit models with a predetermined and dense set of knot points [53, §5.4]. *Adaptive regression splines* [79], on the other hand, use an ℓ_1-type penalty, which can result in a sparse set of adaptively chosen knots. However, adaptive regression splines do not reduce to lower degree outside of the range of their largest knots, and hence may suffer from instability.

We propose fitting adaptive regression splines with explicit constraints on the degree of the spline outside of a certain interval. We call such splines *saturating splines*. While the approach we take can be extended to fitting splines of arbitrary degree with constraints on arbitrary derivatives, in this chapter we focus on fitting linear splines that are flat (constant) outside the data range; we mention the extension to higher degree splines in §4.8. We show that saturating splines inherit the knot-selection property of adaptive regression splines, while at the same time behave like natural splines near the boundaries of the data.

We also show a very important benefit of our approach in the context of fitting generalized additive models [52] with saturating spline coordinate functions: the saturation constraint naturally results in variable selection. Not only do we control the complexity of each coordinate function through knot selection, but with the saturation condition, no knots on a variable means the variable is out of the model. This is not true for adaptive splines, since the linear term is unpenalized and hence each variable would always be in the model. The lack of feature selection can hurt interpretability and, in certain cases, generalization. The saturation constraint we propose precludes linear functions, and in concert with the adaptive spline ℓ_1 penalty encourages coordinate functions to be identically zero. As a result, generalized additive models fit with saturating spline component functions often depend on only a few input features.

Like smoothing splines and adaptive regression splines, saturating splines arise as solutions to certain natural functional regression problems. We solve the saturating spline fitting problem by reformulating it as a convex optimization problem over a space of measures, roughly speaking, the second derivative of the fitted function. To the best of our knowledge, this approach is novel. We then apply a variant of the classical conditional gradient method [60, 14] to this problem. At each iteration of our algorithm, an atomic measure is produced; moreover, we can uniformly bound the number of atoms, which corresponds to the number of knot points in the spline function. (While we manipulate atomic measures, we solve the problem over the space of all measures with finite total variation.) In contrast to standard coordinate descent methods, in each iteration of the conditional gradient method the weights of *two* knot points are adjusted. In the fully corrective step, we solve a finite-dimensional convex optimization problem with ℓ_1 and simple linear constraints. Numerical experiments show that the method is extremely effective in practice.

Our optimization method can exploit warm starts, i.e., it can use an initial guess for the fitted function. This allows us to compute an entire regularization path efficiently, at a cost typically just a small multiple of the effort to solve the problem for one value of the regularization parameter. Because our algorithm is based on the conditional gradient method, we can use the framework of [45] to compute a provably ϵ-suboptimal approximate regularization path. When fitting generalized additive models, the regularization path has attractive features: at critical values of the regularization parameter, new regressors are brought into (or, occasionally, out of) the model, or new knot points are added to (or deleted from) one of the existing coordinate functions. Thus our approach combines feature selection and knot point selection.

Outline

In §4.2 we introduce a univariate function fitting problem, inspired by the adaptive spline estimation problem of [79], that includes the additional requirement that the fitted function saturate. In §4.3 we make the connection between our function estimation problem and standard adaptive splines, and pose the saturating spline fitting problem as a convex optimization problem over measures. In §4.4 we modify the classical conditional gradient method to solve this optimization problem. In §4.5 we extend the optimization problem and algorithm to fit generalized additive models to multivariate data. We illustrate the effectiveness of the method with several examples in §4.7. We discuss generalizations to higher-degree splines in §4.8. Finally, we discuss potential extensions and variations in §4.9. The appendix includes implementation details and proofs.

4.2 Univariate function fitting

We wish to fit a continuous bounded function $f : \mathbf{R} \to \mathbf{R}$ from data $(x_i, y_i) \in \mathbf{R} \times \mathcal{Y}$, $i = 1, \ldots, n$, $x_i \in [0, 1]$. To do this we will choose f to minimize a data mismatch or loss function subject to a constraint that encourages regularity in f, and an additional constraint, saturation, that we describe below.

The loss is given by

$$L(f) = \sum_{i=1}^{n} \ell(f(x_i), y_i),$$

where $\ell : \mathbf{R} \times \mathcal{Y} \to \mathbf{R}$ is nonnegative, twice differentiable, and strictly convex in its first argument. Typical loss functions include $\ell(z, w) = (z - w)^2/2$ (standard regression, $\mathcal{Y} = \mathbf{R}$), or $\ell(z, w) = \log(1 + \exp -(zw))$ (logistic regression, with $\mathcal{Y} = \{-1, 1\}$). The loss L is a convex functional of the function f that only depends on the values of f at the data points x_i. The smaller the loss, the better f fits the given data.

We constrain the function f to be simple by limiting the value of a nonnegative regularization functional R. In this chapter, we take R to be the total variation of the derivative of f,

$$R(f) = \mathrm{TV}(f'),$$

a convex functional of f. For a twice-differentiable function f, recall that

$$\mathrm{TV}(f') = \int |f''(x)| \, dx, \tag{4.1}$$

i.e., the regularization is the ℓ_1 norm of the second derivative. (As we review in the following section, the modern definition of total variation extends this equality to nondifferentiable functions.) The total variation limit we impose on f is $R(f) \le \tau$, where τ is a parameter that we use to trade off model fit and model regularity. This regularization constraint implicitly constrains f to be differentiable almost everywhere, with its derivative having finite total variation.

Our model f will be subject to one more constraint, that it saturates (outside the interval $[0, 1]$), which means that it is a (possibly different) constant on the two intervals outside $[0, 1]$: $f(x) = f(0)$ for $x \leq 0$, and $f(x) = f(1)$ for $x \geq 1$. In other words, f extends as a constant outside the nominal data range of $[0, 1]$. In terms of the derivative, this is equivalent to the requirement that f' exists and is zero outside $[0, 1]$.

The fitting problem is then

$$\begin{array}{ll} \text{minimize} & L(f) \\ \text{subject to} & R(f) \leq \tau, \\ & f'(x) = 0 \text{ for } x \notin [0,1], \end{array} \tag{4.2}$$

where $\tau \geq 0$ is the regularization parameter. The variable to be determined is the function f, which is in the vector space of continuous functions with derivatives of finite total variation. This fitting problem is an infinite-dimensional convex optimization problem.

In applications the problem (4.2) is solved for a range of values of τ, which yields the regularization path. The final model is selected using a hold-out set or cross-validation. For $\tau = 0$, f must be constant and the problem (4.2) reduces to fitting the best constant to the data. As τ increases, f is less constrained, and our fitted model becomes more complex; eventually we expect overfitting. For example, in the case of regression, with a loss function that satisfies $\ell(u, u) = 0$ and data with distinct x_i, the fitting function is the piecewise-linear function that interpolates the data, for large enough τ.

4.3 Splines and functions of bounded variation

In this section we explore the connection between our fitting problem and degree-one splines, i.e., piecewise-linear continuous functions, which have the form

$$f(x) = c + \sum_{i=1}^{K} w_i (x - t_i)_+, \tag{4.3}$$

where $(z)_+ = \max\{z, 0\}$. We assume that the t_i are distinct, and refer to them as knot points or simply knots. The scalars w_i are the weights, and c is the offset. We refer to the function $x \mapsto (x - t_i)_+$ as a hinge function, so a degree-one spline is a finite linear combination of hinge functions, plus a constant.

Functions of bounded variation

A right-continuous function $h : [0,1] \to \mathbf{R}$ is of bounded variation if and only if there exists a signed measure μ on $[0,1]$ with

$$h(z) = \int 1(y \leq z)\, d\mu(y), \tag{4.4}$$

where $1(y \leq z) = 1$ for $y \leq z$ and 0 otherwise. The measure μ is unique; we can think of it as the derivative of h. That is, (4.4) is essentially the second fundamental theorem of calculus with h' replaced by μ.

We also have $\mathrm{TV}(h) = \int d|\mu|$. (This is called the total variation of the measure μ.) We will denote this using the notation $\|\mu\|_1$, to emphasize the similarity with the finite-dimensional case, or the case when h is differentiable: $\mathrm{TV}(h) = \|h'\|_1$. When the measure μ is atomic, the function h is piecewise constant with jumps at the points in the support of μ.

Splines and derivatives with bounded variation

Now suppose that $f : [0,1] \to \mathbf{R}$ has a right-continuous *derivative* of bounded variation. From (4.4), with $h = f'$, and the fundamental theorem of calculus, we have

$$\begin{aligned} f(x) &= f(0) + \int_0^x f'(z)\, dz = f(0) + \int_0^x \int 1(y \leq z)\, d\mu(y)\, dz && (4.5)\\ &= f(0) + \int \int_0^x 1(y \leq z)\, dz\, d\mu(y) && (4.6)\\ &= f(0) + \int (x-y)_+\, d\mu(y). && (4.7)\end{aligned}$$

This shows that any such function is a (possibly infinite) linear combination of hinge functions, plus a constant (i.e., $f(0)$). In this case, the measure μ can be thought of as the *second* derivative of f.

When μ is atomic and supported on a finite set, that is,

$$\mu = \sum_{i=1}^{K} w_i \delta_{t_i},$$

f is a degree-one spline of the form (4.3), with $c = f(0)$. So degree-one splines correspond exactly to the case where the measure μ (roughly, the second derivative) has finite support.

We introduce the notation

$$f_\mu(x) = \int_0^x \int 1(t \leq z)\, d\mu(t)\, dz = \int (x-t)_+\, d\mu(t) \tag{4.8}$$

to denote the function derived from the measure μ. It is, roughly speaking, the double integral of the measure μ, or the (potentially infinite) linear combination of hinge functions associated with the measure μ. The mapping from μ to f_μ is linear, and we have $\mathrm{TV}(f'_\mu) = \|\mu\|_1$. A simple example of f_μ, its first derivative f'_μ, and its (atomic measure) second derivative μ is shown in Figure 4.1.

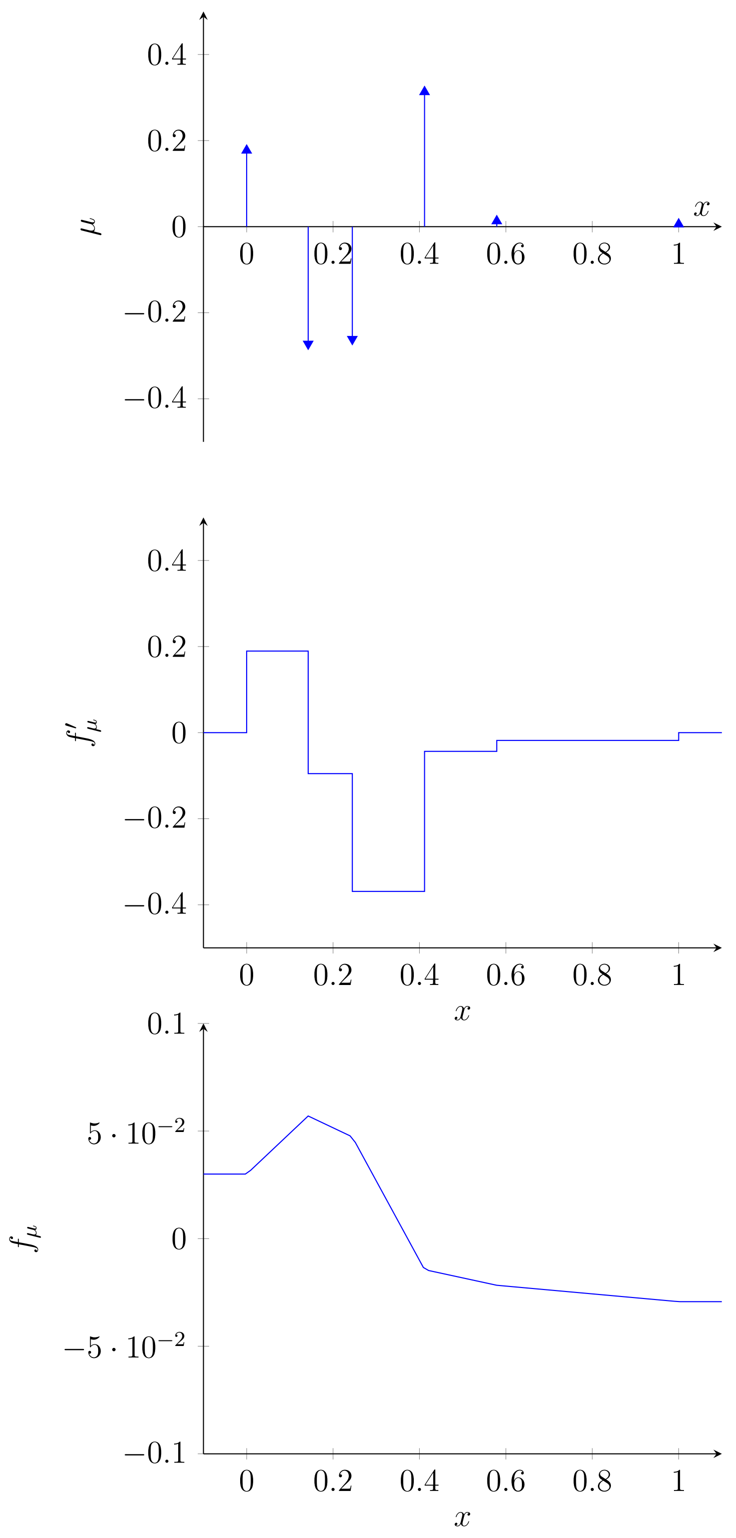


Figure 4.1: f_μ and f'_μ generated by the atomic measure μ (f''_μ). The regularization functional, $\mathrm{TV}(f'_\mu)$, is the sum of the absolute values of the spikes in μ. Note that the (signed) sum of the spikes in μ is zero: that is, $\int d\mu = 0$, which implies that f_μ saturates.

Fitting splines by optimizing over measures

Identifying $f = c + f_\mu$, we can solve the fitting problem (4.2) by minimizing over the bounded measure μ on $[0, 1]$, and the constant c. The measure μ is the second derivative of f, and the constant c corresponds to $f(0)$. The total variation regularization constraint $\mathrm{TV}(f') \leq \tau$ corresponds to $\|\mu\|_1 \leq \tau$. The saturation condition holds by construction for $x < 0$; to ensure that $f'(x) = 0$ for $x > 1$, we need

$$f'(1) = f'(0) + \int_0^1 d\mu = 0.$$

In other words, saturation of f corresponds to μ having total (net) mass zero. Thus (4.2) can be rephrased as

$$\begin{array}{ll} \text{minimize} & L(E_x\mu + c) \\ \text{subject to} & \|\mu\|_1 \leq \tau, \\ & \int d\mu = 0 \end{array} \tag{4.9}$$

over the bounded measure μ on $[0, 1]$, and $c \in \mathbf{R}$. Note the slight abuse of notation here: we now (and for the remainder of the chapter) consider L as a functional on $\mathbf{R}^n$. In the above, E_x is the linear operator that maps μ to the vector $(f_\mu(x_1), \ldots, f_\mu(x_n))$, given by (4.8). E_x is clearly linear, as it is the integral of the function $\phi : \mathbf{R} \to \mathbf{R}^n$:

$$\phi(t) = ((x_1 - t)_+, \ldots, (x_n - t)_+)$$

against μ. We will apply the conditional gradient method directly to this problem.

To gain intuition about the optimization problem (4.9), we can consider it as a infinite-dimensional analogue of the standard lasso [115]. The lasso is the solution to the optimization problem

$$\begin{array}{ll} \text{minimize} & \frac{1}{2}\|Aw - y\|_2^2 \\ \text{subject to} & \|w\|_1 \leq \tau. \end{array} \tag{4.10}$$

Here w is a vector in $\mathbf{R}^d$, and $A \in \mathbf{R}^{(n,d)}$ is a matrix. Ignoring the constant term c, we see that (4.9) looks very similar to (4.10), where E_x plays the role of A; indeed, E_x is essentially a matrix with n rows and infinitely many columns. Our intuition from the lasso suggests that there should be solutions of (4.9) that are sparse, which here means that μ is atomic. In terms of f_μ, sparsity means there are solutions of the original functional fitting problem (4.2) that are degree-one splines. This is indeed the case. Theorem 6 shows that there is a solution of (4.9) with μ atomic, supported on no more than $n + 2$ points; in other words, f_μ is a degree-one spline with $K \leq n + 2$. Moreover, in practice the solution of (4.9) will exhibit selection, that is, it will be supported on far fewer than $n + 2$ points.

Theorem 6. *Fix $x_1, \ldots, x_n \in [0, 1]$ and $f : \mathbf{R} \to \mathbf{R}$ with f' (right-continuous) of bounded total variation, and f constant outside of $[0, 1]$. Then there exists a degree-one saturating spline $\hat{f}$ (with an most $n + 2$ knots) that matches f on x_i with $\mathrm{TV}(\hat{f}') \leq \mathrm{TV}(f')$.*

For the remainder of the chapter we will ignore the constant term c. It is not difficult to adapt the algorithms we present to handle the constant term, but doing so does add some notational complexity. It's also possible to minimize out c, as it does not affect the regularization term; the resulting problem is still convex in w.

4.4 The conditional gradient method for fitting splines

In this section we outline our algorithm for solving (4.9) (and therefore also (4.2)). To that end, we briefly review the classical conditional gradient method [60] and the measure-theoretic version proposed in [14].

The optimization problem we need to solve, (4.9), (without the constant term c) is

$$\begin{array}{ll} \text{minimize} & L(E_x\mu) \\ \text{subject to} & \int d\mu = 0, \\ & \|\mu\|_1 \leq \tau. \end{array} \tag{4.11}$$

As noted in the last section, (4.11) is a convex optimization problem over a space of measures. We closely follow the approach taken in [14] and apply the conditional gradient method to this problem directly.

The main benefit of this approach is that we can restrict our attention to *atomic* measures, i.e., μ of the form

$$\mu = \sum_{j=1}^{K} w_j \delta_{t_j}.$$

Measures of this form are easily representable in a computer, by simply storing a list of (w_j, t_j) pairs. Theorem 6 ensures that the number of knots we need to store is absolutely bounded, i.e., that our algorithm runs in bounded memory. While we manipulate atomic measures, we solve the problem (4.11) over all bounded measures.

One thing to note about finitely-supported atomic measures is that we can easily optimize over the weights w_j with the knot locations t_j fixed, since this corresponds to a finite-dimensional convex optimization problem amenable to any standard algorithm. Our algorithm makes use of this fact, and alternates between adding pairs of knots and optimizing over the weights w at each iteration. In this latter step knots can be (and indeed eventually must be) removed. In an additional and optional step the knot points can be moved continuously within $[0, 1]$, or to neighboring data points. As we saw in Chapter 2, this step is not needed for theoretical convergence but can improve convergence and the sparsity of the final solution in practice.

The conditional gradient method

The conditional gradient method (CGM) solves constrained convex optimization problems of the form

$$\begin{array}{ll} \text{minimize} & f(x) \\ \text{subject to} & x \in \mathcal{C}, \end{array} \tag{4.12}$$

with variable $x \in \mathbf{R}^d$. In the above, it is always assumed that the (convex) function f is differentiable. At each iteration of the CGM we form the standard linear approximation to the function f at the current iterate x_m:

$$\hat{f}(x; x_m) = f(x_m) + f'(x - x_m; x_m).$$

Here $f'(d; x)$ is the directional derivative of the function f at x in the direction d, defined by

$$f'(d; x) = \lim_{t \searrow 0} \frac{f(x + td) - f(x)}{t}.$$

Our use of the directional derivative here may seem surprising: for differentiable functions on $\mathbf{R}^d$, $f'(d; x)$ is always equal to $\langle \nabla f(x), d \rangle$. The direct applicability of directional derivatives to convex functionals of measures motivates us to prefer the directional derivative.

Convexity of f implies that $\hat{f}$ is a *lower* bound on f, that is:

$$\hat{f}(x; x_m) \leq f(x). \tag{4.13}$$

In the next step of the CGM, we minimize this first-order approximation over the feasible set $\mathcal{C}$:

$$s_m \in \arg\min_{s \in \mathcal{C}} \hat{f}(s; x_m) = \arg\min_{s \in \mathcal{C}} f'(s; x_m).$$

The point s_m is called the conditional gradient of f. Note that s_m provides a lower bound on $f(x_\star)$:

$$\hat{f}(s_m; x_m) \leq f(x_\star).$$

In particular, we can bound the sub-optimality of the point x_m:

$$f(x_m) - f(x_\star) \leq -f'(s_m - x_m; x_m). \tag{4.14}$$

One can show [60] that this bound decreases to zero, which means that it can be used as a (non-heuristic) termination criterion. After determining s_m, there are several options for updating x_m. In this chapter, we will use the fully-corrective variant of the CGM, which chooses x_{m+1} to minimize f over the convex hull of $\{s_1, s_2, \ldots, s_m\}$. Note that this last step may become computationally intensive as k grows, and indeed limits the applicability of the conditional gradient method to problems where this step is computationally feasible. One option is to remove previous conditional gradients as soon as they are not selected in the minimization step. Caratheodory's theorem ensures that the set of previous conditional

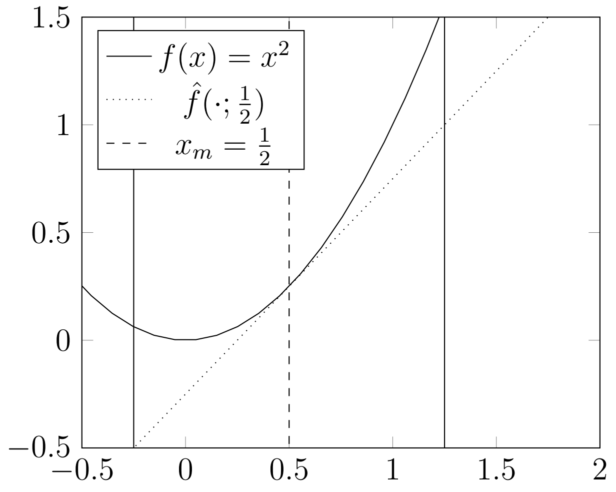


Figure 4.2: An illustration of a single iteration of the conditional gradient method on the function $f(x) = x^2$ at the point $\frac{1}{2}$. The set $\mathcal{C}$ is the interval $[-0.25, 1.25]$, indicated by the solid vertical lines. The first order approximation $\hat{f}(\cdot; \frac{1}{2})$ is plotted as the dotted line tangential to $f(x)$ at $\frac{1}{2}$. The conditional gradient s_m is the point -0.25.

gradients we need to track is then bounded by $d+1$. In practice, however, the algorithm is usually terminated well before $d+1$ iterations.

Algorithm 4 Fully-corrective conditional gradient method

```
For $m = 1, \ldots$
    1. Linearize: $\hat{f}(s; x_m) \leftarrow f(x_m) + f'(s - x_m; x_m)$.
    2. Minimize: $s_m \in \arg\min_{s \in \mathcal{C}} \hat{f}(s; x_m)$.
    3. Update: $x_m \in \arg\min_{x \in \text{conv}(s_1, \ldots, s_m)} f(s)$.
```

Conditional gradient for measures

In this subsection, we apply the conditional gradient method to the infinite-dimensional problem (4.11), which we repeat here:

$$\begin{array}{ll} \text{minimize} & L(E_x \mu) \\ \text{subject to} & \int d\mu = 0, \\ & \|\mu\|_1 \le \tau. \end{array} \tag{4.15}$$

First we'll show that the conditional gradient, i.e., the measure s_m, can be chosen to be supported on exactly two points, and is computable in time linear in n. The directional derivative of the objective function in the direction of the measure s at the point μ is given by

$$\begin{aligned}
&\lim_{t \searrow 0} \frac{L(E_x(\mu + ts)) - L(E_x\mu)}{t} \\
&= \lim_{t \searrow 0} \frac{L(E_x\mu + tE_x s) - L(E_x\mu)}{t} \\
&= L'(E_x s; E_x\mu) \\
&= \langle \nabla L(E_x\mu), E_x s\rangle_{\mathbf{R}^n}.
\end{aligned}$$

We can then interchange the inner-product in $\langle \nabla L(E_x\mu), E_x s\rangle$ with the integral in $E_x s = \int \phi(t)\, ds(t)$:

$$\langle \nabla L(E_x\mu), E_x s\rangle = \int \langle \nabla L(E_x\mu), \phi(t)\rangle\, ds(t). \tag{4.16}$$

Let $g = \nabla L(E_x\mu) \in \mathbf{R}^n$. Note that in the case $\ell(x, y) = \frac{(x-y)^2}{2}$, g is simply the residual $E_x\mu - y$ and $\langle g, \phi(t)\rangle$ is the correlation between the residual and a single hinge function located at t. A conditional gradient is any solution to the following optimization problem

$$\begin{array}{ll}
\text{minimize} & \int \langle g, \phi(t)\rangle\, ds(t) \\
\text{subject to} & \int ds = 0, \\
& \|s\|_1 \leq \tau.
\end{array} \tag{4.17}$$

Without the integral constraint, we would expect there to be a solution to (4.17) that is a single point-mass: the objective function is the integral of a scalar-valued function against a bounded measure. We'll show that there is always a solution to (4.17) that is supported on exactly two points. Furthermore, we'll show that those two points can be computed in time linear in n.

First we'll construct a particular feasible point for (4.17) and then we'll show that it achieves the optimal value. Let

$$t_+ \in \arg\min_t \langle g, \phi(t)\rangle, \quad t_- \in \arg\min_t -\langle g, \phi(t)\rangle.$$

Define

$$s_\star = \frac{\tau}{2}\delta_{t_+} - \frac{\tau}{2}\delta_{t_-}.$$

The objective value achieved by $s_\star$ is

$$o_\star = \frac{\tau}{2}\left(\langle g, \phi(t_+)\rangle - \langle g, \phi(t_-)\rangle\right).$$

We'll show that either any measure s that is feasible for (4.17) has objective value bounded below by $o_\star$ or μ is optimal for (4.11). Let s be any feasible measure for (4.17). Decompose s into the difference of two mutually singular non-negative measures: $s = s_+ - s_-$. Then as

s is feasible we have $\|s_+\|_1 = \|s_-\|_1 \leq \frac{\tau}{2}$. The objective value achieved by s can be bounded below as follows

$$\begin{aligned}\int \langle g, \phi(t)\rangle ds(t) &= \int \langle g, \phi(t)\rangle ds_+(t) + \int -\langle g, \phi(t)\rangle ds_-(t) \\ &\geq \|s_+\|_1 \left(\min_t \langle g, \phi(t)\rangle\right) + \|s_-\|_1 \left(\min_t -\langle g, \phi(t)\rangle\right) \\ &\geq \|s_+\|_1 \left(\min_t \langle g, \phi(t)\rangle + \min_t -\langle g, \phi(t)\rangle\right).\end{aligned}$$

Suppose $(\min_t\langle g, \phi(t)\rangle + \min_t -\langle g, \phi(t)\rangle) \geq 0$. Then the argument above implies $s_\star = 0$ is a conditional gradient for (4.11), and thus (3.4) implies μ is optimal. Otherwise we have

$$\left(\min_t \langle g, \phi(t)\rangle + \min_t -\langle g, \phi(t)\rangle\right) < 0,$$

which implies

$$\|s_+\|_1 \left(\min_t \langle g, \phi(t)\rangle + \min_t -\langle g, \phi(t)\rangle\right) \geq \frac{\tau}{2}\left(\min_t \langle g, \phi(t)\rangle + \min_t -\langle g, \phi(t)\rangle\right) = o_\star.$$

This proves the assertion.

Note that finding t_- and t_+ involves two *separate* optimization problems over $[0,1]$ instead of one over $[0,1]\times[0,1]$. These problems are readily solved by gridding, though in this case they can be solved exactly in time linear in n if we have access to a sorted vector of the data points x_i. To see this, we expand the objective function for t_+ above,

$$t_+ = \arg\min_{0\leq t\leq 1} \sum_{i=1}^n g_i(x_i - t)_+ = \arg\min_t \sum_{i:x_i\geq t} g_i(x_i - t).$$

If x_i are sorted, we can compute the minimizer between each pair of consecutive data points exactly, since this is simply computing the minimizer of a linear functional over an interval. Thus in a single pass over the data we can compute the global minimizer exactly.

Immediately after computing t_- and t_+ we can use (3.4) to bound the suboptimality of μ by

$$L(E_x\mu) - L(E_x\mu_\star) \leq -\int \langle g, \phi(t)\rangle d(s_\star - \mu)(t).$$

With this choice of conditional gradient, the fully-corrective step is a finite-dimensional convex problem. Fixing the knot locations encountered as conditional gradients so far, $t_1, \ldots, t_{2k}$, we can do at least as well as the fully-corrective algorithm by solving the following optimization problem:

$$\begin{array}{ll} \text{minimize} & L(E_x\mu) \\ \text{subject to} & \int d\mu = 0, \\ & \|\mu\|_1 \leq \tau, \\ & \text{supp}(\mu) \subset \{t_1, \ldots, t_{2k}\}. \end{array} \tag{4.18}$$

This is equivalent to the following optimization problem in $\mathbf{R}^{2k}$:

$$\begin{array}{ll} \text{minimize} & L(\sum_j w_j E_x \delta_{t_j}) \\ \text{subject to} & 1^T w = 0, \\ & \|w\|_1 \leq \tau. \end{array} \tag{4.19}$$

We can solve this using any of a number of existing algorithms [15, 118]. In our implementation we use the conditional gradient method with line-search for simplicity.

By warm starting with an increasing sequence of τ's, we can efficiently compute an approximate regularization path. Indeed we can even provide a provably ϵ-suboptimal path using the approach of [45].

Convergence

As in the case of ADCG [14] convergence follows immediately from the conditional gradient method proof in general Banach spaces [35, 30, 60]. The convergence of the conditional gradient method depends on a curvature parameter C_f. C_f is a constant such that the following inequality is satisfied for all $x, s \in \mathcal{S}$ and $\eta \in (0, 1)$:

$$f(x + \eta(s - x)) \leq f(x) + \eta f'(s - x; x) + \frac{C_f}{2}\eta^2.$$

For our purposes $f : \mathbf{R}^n \to \mathbf{R}$ is simply L and $\mathcal{S} = \{E_x\mu : \|\mu\|_1 \leq \tau, \int d\mu = 0\}$. A simple sufficient condition for C_f to be finite is that ℓ is differentiable with Lipschitz gradient. If C_f is finite, the conditional gradient method converges (in terms of function value) at a rate of at least $1/m$ where m is the iteration counter.

4.5 Generalized additive models

One natural application of univariate splines is fitting generalized additive models [52] to multivariate data: $(x_i, y_i) \in \mathbf{R}^D \times \mathcal{Y}$, $i = 1, \ldots, n$. That is, fitting a function of the form

$$f(x) = \sum_{d=1}^{D} f_d(x[d])$$

where each f_d is a simple function from $\mathbf{R}$ to $\mathbf{R}$ (here $x[d]$ is the d-th coordinate of the vector x). We can mimic our approach in the scalar case with the following optimization problem:

$$\begin{array}{ll} \text{minimize} & L(f) \\ \text{subject to} & \sum_d R(f_d) \leq \tau, \\ & f_d'(x) = 0 \ \forall x \notin [0, 1], d. \end{array} \tag{4.20}$$

Here R is the same regularizer used in the scalar case, namely

$$R(g) = \mathrm{TV}(g') \simeq \|g''\|_1.$$

As in the scalar case, one can show that there is always an optimal f with each coordinate function f_d a degree-one saturating spline.

This allows us to rephrase (4.20) as an optimization problem over measures. The only change from the scalar case is that the measure is over the set $\{1, \dots, D\} \times [0, 1]$—each knot is now attached to a particular coordinate. In other words, we search for a function of the following form:

$$f_\mu(x) = \int (x[d] - t)_+ d\mu(d, t).$$

We again have equality between the ℓ_1 norm of μ and the regularization term:

$$\sum_d R((f_\mu)_d) = \|\mu\|_1.$$

The analogue of (4.11) is then

$$\begin{array}{ll} \text{minimize} & L(E_x \mu) \\ \text{subject to} & \int 1(d = \hat{d})\ d\mu(d, t) = 0,\ \forall \hat{d} \\ & \|\mu\|_1 \leq \tau. \end{array} \tag{4.21}$$

The conditional gradient algorithm from the scalar case generalizes immediately to fitting generalized additive models—the only difference is that we now need to find a pair of knots for the same coordinate. This involves solving d pairs of nonconvex optimization problems over $[0, 1]$—again this can be done by gridding or by sorting the training data.

Saturating splines gain an additional advantage over standard adaptive splines when fitting generalized additive models. The addition of the saturation constraint (that f_d be constant outside of $[0, 1]$) naturally leads to variable selection when fitting generalized additive models. What we mean by variable selection is that the functions f_d are often exactly 0. This is because the saturation constraint means that linear coordinate functions no longer escape the regularization (indeed, they are impossible). This is very different from the standard adaptive spline setup without the saturation constraint. In that case, linear functions, i.e., $f_d(x[d]) = wx[d]$ completely escape the regularization, and as a result are essentially always included in the model. Linear functions are *not* free with saturation constraints (in fact, outside of the function 0, they are not feasible). When we solve (4.21) we simultaneously fit nonlinear coordinate functions while doing variable selection.

4.6 Prior and related work

Smoothing splines also have an interpretation as the solution of an infinite-dimensional optimization problem [53, §5.4]. In fact, (degree-one) smoothing splines solve

$$\begin{array}{ll} \text{minimize} & L(f) \\ \text{subject to} & \hat{R}(f) \leq \tau, \end{array} \tag{4.22}$$

where

$$\hat{R}(f) = \int f'(x)^2 \, dx.$$

The solution to (4.22) is *also* a degree-one natural spline that saturates outside of $[0, 1]$. However, the solutions to (4.22) and (4.2) are *very* different. Roughly, (4.22) is analogous to ridge regression, while (4.2) is analogous to the lasso. That is, (4.22) fits functions with as many knots as datapoints, while (4.2) often fits splines with very few knots.

Another type of spline, that is adaptive but does not saturate, are adaptive regression splines [79]. These splines also arise as solutions to a functional regression problem:

$$\begin{array}{ll} \text{minimize} & L(f) \\ \text{subject to} & \hat{R}(f) \leq \tau, \end{array} \tag{4.23}$$

where

$$\hat{R}(f) = TV(f'(x)).$$

Note that this is (4.2) without the saturation constraint. Algorithms for solving (4.23) (for degree-one splines) are based on an extension of Theorem 6, that shows there is a solution to (4.23) which is actually supported on the data points x_i. Hence a lasso algorithm can be used to find the solution. This also suggests a very simple method to solve our problem (4.9): we fix the n knot points to be the values of the data x_i, and solve the finite-dimensional convex optimization problem to find the weights. While simple coordinate-descent methods like GLMNet [43] will not immediately work because of the saturation constraint, they could be modified to handle the constraint.

This method does work, but can be much slower than ours since in practice the number of knots is typically much smaller than n for useful values of the regularization parameter τ, and the finite-dimensional problem with n basis functions is very poorly conditioned. With that said, the algorithm we propose—for the piecewise linear case—can be interpreted as a forward active set method for the finite dimensional problem, where we avoid explicitly evaluating all basis functions. One advantage of our measure-theoretic approach is that it immediately generalizes to higher-degree splines, where the support of μ need not be on data points, as we will see in §4.9. In this case (4.9) is truly infinite-dimensional, yet our algorithm can still be directly applied.

Trend filtering is a nonparametric function estimation technique, first introduced by [67], that is very similar to adaptive splines. Indeed, as discussed by [114], the trend filtering estimate in the constant or piecewise-linear case is exactly the same as the adaptive spline estimate. Trend filtering is increasingly popular as it admits extremely efficient, robust algorithms [114, 94]. Indeed, some of these algorithms (especially those adapted to fit GAMs, [89]) may be adapted to efficiently fit saturating trend filter estimates, which would benefit from the feature selection properties of saturating splines and the computational efficiency of trend filtering.

There are a number of methods for fitting generalized additive models with spline component functions. One approach [75] is to use the group-lasso version of (4.6):

$$R(f) = \sum_d \sqrt{\int f_d'(x)^2 \, dx}.$$

Extending this idea, [24] use an overlap group-lasso that facilitates selection between zero, linear and nonlinear terms. The differences between these approaches and ours are analogous to the differences between the standard group-lasso and the lasso. While both do feature selection, the penalty functional (4.6) does not do knot-selection within each coordinate function.

One very similar approach to fitting splines that does not require knot selection (but does not incorporate saturation) is discussed by [100].

4.7 Examples

In all examples we affinely preprocess the data so that all training features lie in $[0, 1]$, and apply the same transformation to the test features (which thus may have values outside of $[0, 1]$). All plots are in terms of the standardized features. For the bone density and abalone data sets we select τ to minimize error on the validation sets. For the Spam and ALS data sets we use cross-validation to estimate τ. We hold out a random subset of size 100 from the training set and train on the remaining data. For each random validation/train split we estimate τ to minimize hold-out error and take our final estimate of τ as the mean over 50 trials.

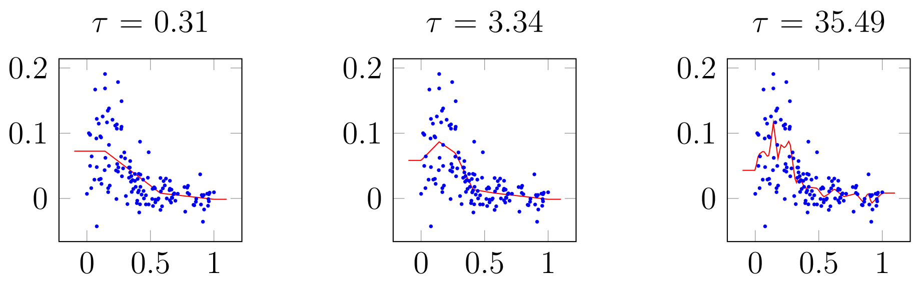


Figure 4.3: Saturating splines fit to bone density data (shown as scattered points) for 3 values of the regularization parameter τ. *Left:* $\tau = 0.31$; *Middle:* $\tau = 3.34$; *Right:* $\tau = 35.45$.

Bone density

We start with a simple univariate data set from [53, §5.4]. The response variable for this data set is the change in spinal bone density between two doctor visits for female adolescents as a function of age. There are 259 data points, of which we hold out 120 for validation, leaving 139 data points to which we fit a saturating spine. We start with the square loss.

The results are shown in figure 4.3, for three values of the regularization parameter τ. The scattered points are the training data, the solid line is the saturating spline fit by our algorithm. The figure demonstrates the clear link between τ and the complexity of the optimized spline. Out-of-sample validation suggests setting $\tau \simeq 3.34$, which achieves a validation RMSE of 0.036.

To demonstrate that our proposed method works with more general loss functions, we add 30 simulated outliers to the training set and fit with the pseudo-Huber loss [22], a smooth approximation to the Huber loss function given by

$$l_\delta(u) = \delta\left(\sqrt{1 + \frac{u^2}{\delta}} - 1\right),$$

where $\delta > 0$ is a parameter that interpolates between the absolute value loss and the squared loss. For our experiment we take $\delta = 0.0015$; roughly speaking, the transition between square and linear loss occurs around $\sqrt{\delta} = 0.039$. The results are shown in figure 4.4. These plots demonstrate that our algorithm can fit losses other than the square loss, and confirms that the pseudo-Huber loss is far more robust to outliers than the basic square loss function. Indeed, on the validation set the least-squares fit achieves a minimum RMSE of 0.096, while the pseudo-Huber fit achieves 0.038, only slightly worse than the fit obtained before the outliers were added to the training data. While this one-dimensional problem is very easy, it shows one advantage of the adaptive spline penalty over smoothing splines: the optimal model has only 5 knot points.

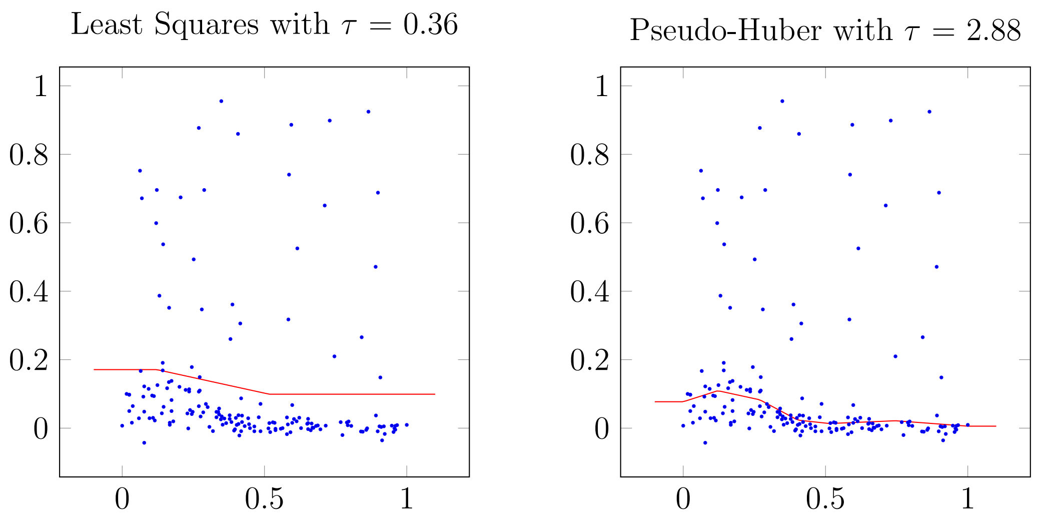


Figure 4.4: Saturating splines fit to bone density data (shown as scattered points) with simulated outliers for square loss function (left) and pseudo-Huber loss function (right), each for the value of τ that minimizes RMSE on the test set.

Abalone

We fit a generalized additive model with saturating spline coordinate functions to the Abalone data set from the UCI Machine Learning Repository [74]. The data consists of 4177 observations of 8 features of abalone along with the target variable, the age of the abalone. We hold out 400 data points as a validation set, leaving 3777 data points to fit the model. The first feature (labeled sex) has three values: Male, Female, and Juvenile, which are coded with values $0, 1, 2$; the other 7 are (directly) real numbers. The task is to estimate the age of the abalone from the features.

Cross-validation suggests we choose $\tau \simeq 200$, which achieves a validation set RMSE of 2.131. Because the number of features is low, we can plot the entire generalized additive model. Each plot shows one coordinate function f_d for $d = 1, \ldots, 8$ as a function of the standardized feature in $[0, 1]$. The coordinate functions are shown for three values of τ, with the middle one corresponding to the value that minimizes cross-validation RMSE. When a coordinate function is zero, which means that the feature is not used in the model, it is shown in blue. We can see that in the case of strong regularization ($\tau = 20$), several coordinates are not used; for the best model ($\tau = 200$), all features are used, with a few having only a small effect. It is interesting to see how the sex factors into the optimal model. It is neutral on Male or Female, but subtracts a small fixed amount from its age prediction for a Juvenile abalone.

This data set is small enough that we can compare against standard adaptive splines fit using a coarse grid of $[0, 1]$. For this experiment, we fit a GAM with standard adaptive spline component functions using GLMNET [43]. The standard adaptive GAM fit, which does no variable selection, achieves a validation set RMSE of 2.137, not significantly worse than the saturating spline model. Our algorithm, however, selects many fewer knot points. The increased number of knots when fitting with GLMNET is perhaps due to the poor conditioning of the gridded problem.

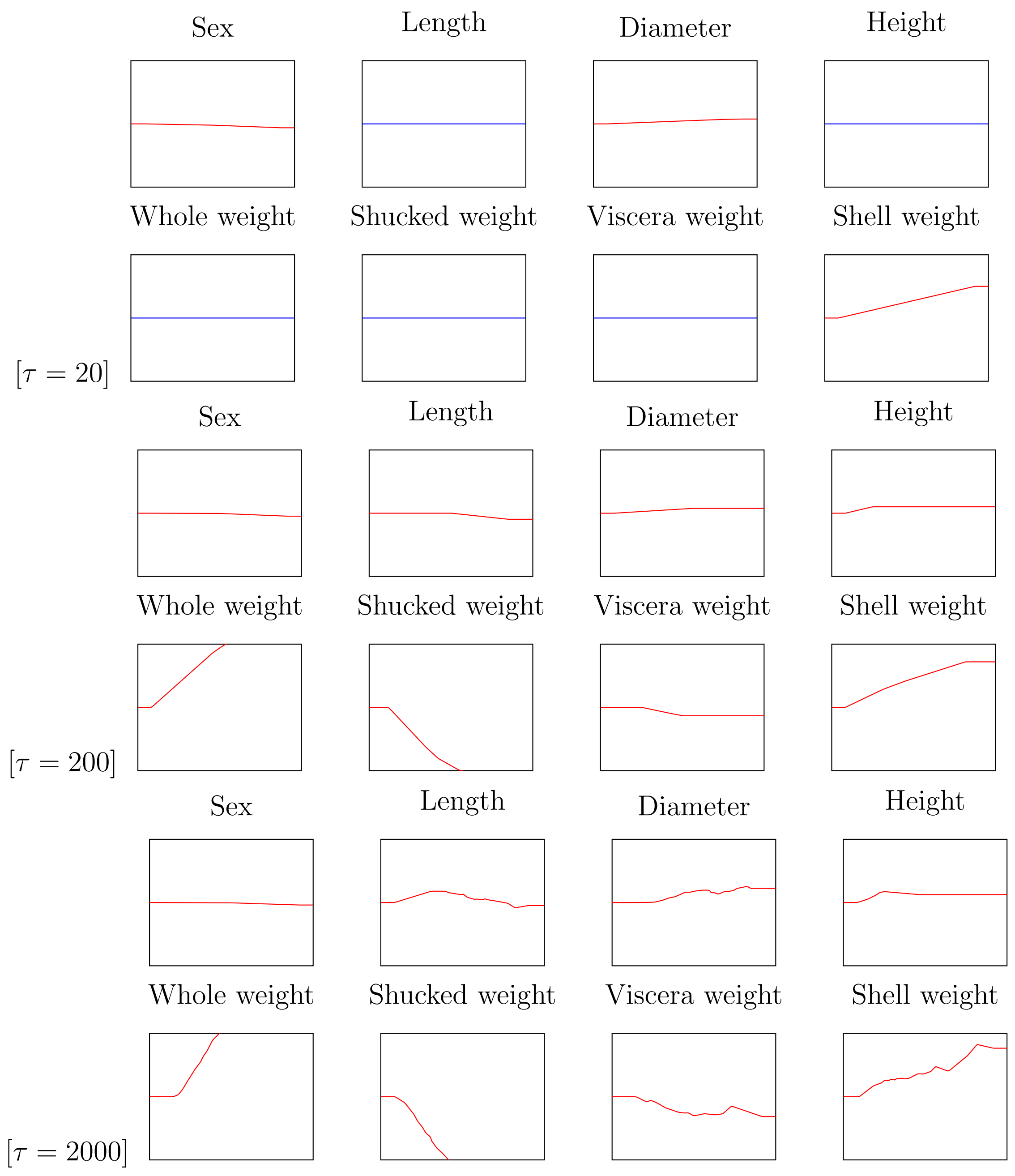


Figure 4.5: Coordinate functions for saturating spline generalized additive models fit to Abalone data for three values of the regularization parameter τ.

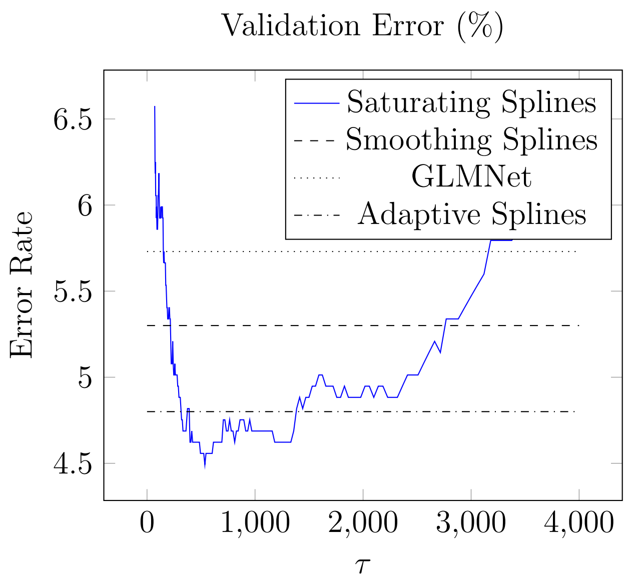


Figure 4.6: Validation error for saturating spline generalized additive model fit to Spam data set versus regularization parameter τ.

Spam

We consider the problem of classifying email into spam/not spam, with a data set taken from ESL [53]. The data set consists of 57 word-frequency features from 4601 email messages, along with their labels as spam or not spam. Following the approach in ESL [53] we log-transform the features and use the standard train/validation split, with a training set of size 3065, and test set with 1536 samples. We fit a saturating spline generalized additive model with standard logistic loss.

Figure 4.6 shows the validation error versus the regularization parameter τ. Cross-validation suggests the choice $\tau \simeq 1100$. To show the benefit of nonlinear coordinate functions, we also include the best validation error achieved using a linear model (fit using GLMNet [43]).

With regularization parameter $\tau = 500$, the model selects 55 of the 57 features. We note that our saturating spline generalized additive model modestly outperforms many methods from ESL [53]; for example, smoothing splines yield 5.3% error, while our model has an error rate well below 5%. Figure 4.7 shows (some of) the coordinate functions for the model with $\tau = 500$. The coordinate functions use very few knots, making them readily interpretable.

For comparison, we fit a GAM with standard adaptive spline coordinate functions. To do so, we grid each dimension with 20 knots and solve the resulting finite-dimensional problem with GLMNET [43]. Note that adaptive splines do not penalize linear functions, so there is no feature selection. Adaptive splines achieve a minimum error of 4.8%, significantly worse than saturating splines.

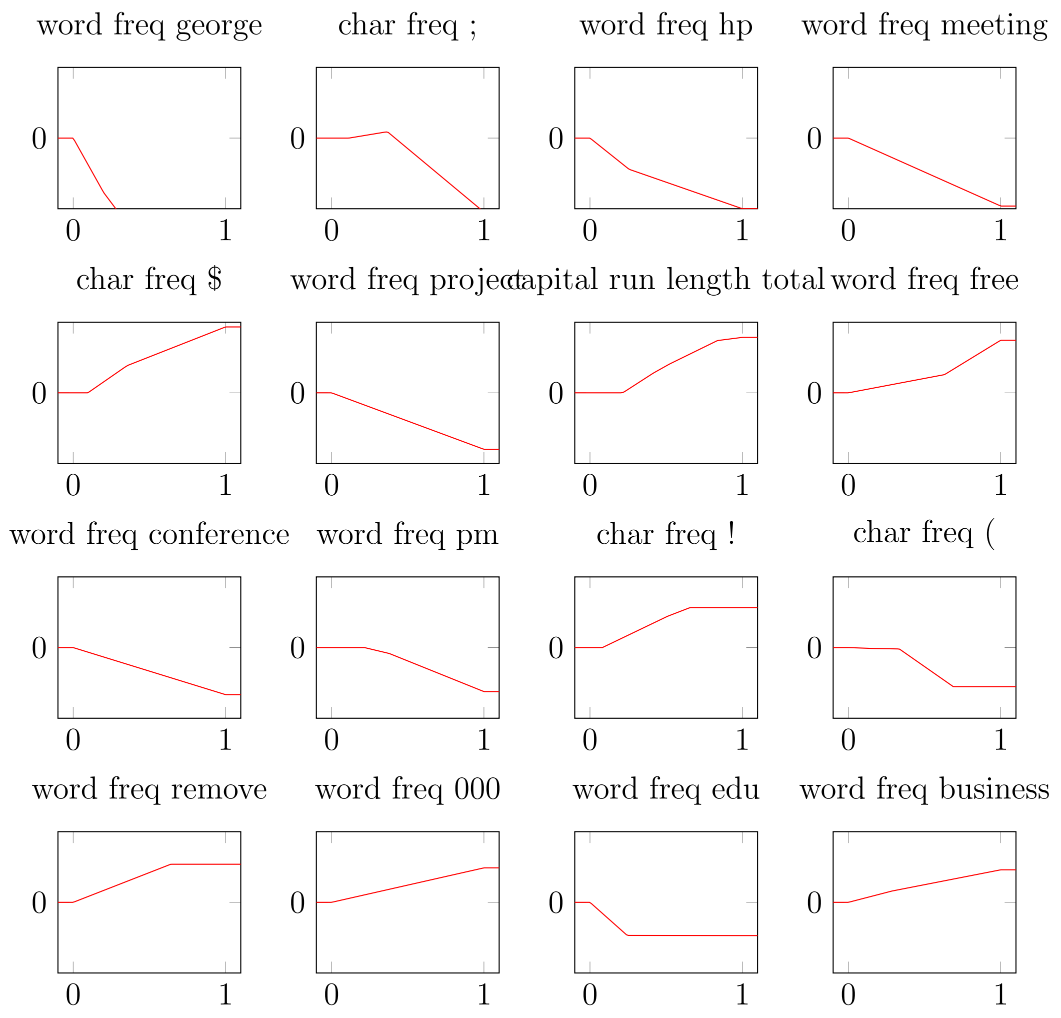


Figure 4.7: 16 coordinate functions for $\tau = 500$, labeled with the corresponding feature name.

ALS

Using this data set we try to predict the rate of progression of ALS (amyotrophic lateral sclerosis) in medical patients, as measured by the rate of change in their functional rating score, a measurement of functional impairment. The data set is split into a training set of 1197 examples and a validation set of 625 additional patients. Each datapoint has dimension 369. We fit a generalized additive model with saturating spline component functions to the data using a least-squares objective function. Following [36, §17.2], we measure performance using mean-squared error.

We estimate the optimal value of τ using cross validation with a hold-out size of 100 examples and 50 samples; this procedure suggests $\tau = 13$. Figure 4.8 shows the validation error versus the regularization parameter τ; the value of τ selected by cross validation achives low error. On the same plot, we also show the results from [36] using boosted regression trees and random forests. The optimal saturating spline GAM model selects only 50 out of the 369 features, in contrast to boosted regression trees, which use 267. The saturating spline GAM model performs comparably to boosted regression trees and random forests. This is surprising as the saturating spline GAM has no interaction terms. It also uses substantially fewer features, further improving interpretability.

Again we fit a GAM with standard adaptive spline coordinate functions (using GLMNET) to show the advantage of saturation. The standard adaptive spline fit achieves an MSE of 0.547, substantially worse than any other model. We speculate that this is because the unpenalized linear functions lead to immediate overfitting. Indeed, removing the unpenalized linear functions and fitting a model with only hinges gives very similar performance to the saturating spline fit, suggesting that the main advantage of saturation for this application is the removal of the unpenalized linear functions.

Practical advantages of saturating splines

These experiments show that saturating splines achieve competitive performance on small classification and regression data sets. In addition, the experiments demonstrate that saturating splines exhibit both knot selection and feature selection—in the context of fitting GAMs. While it is no surprise that saturating splines select fewer knots than smoothing splines (which choose a fully-dense set of knots), it is somewhat surprising that our algorithm selects fewer knots than even *adaptive* splines fit with GLMNET. Finally, the Spam and ALS data sets demonstrate a major advantage of saturating splines over adaptive splines: they simultaneously perform non-linear coordinate function fitting and feature selection. This aids in generalization performance and interpretability. In particular, for the ALS data set saturating spline GAMs achieve *half* the test MSE of adaptive spline GAMs by selecting only 50 of 369 available features.

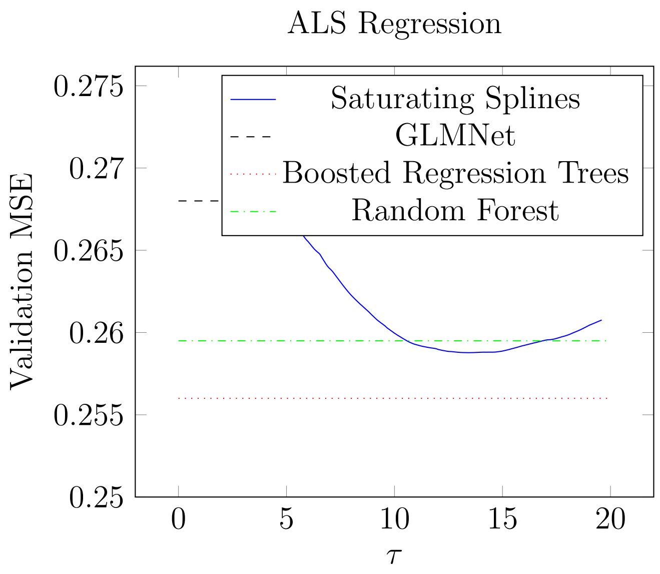


Figure 4.8: Validation MSE on ALS data set versus regularization parameter τ.

4.8 Higher-degree splines

In the majority of this chapter we focused on the functional regression problem (4.2), with a total variation constraint on the first derivative and a saturation constraint on the zeroth derivative (the function itself). In this section, we consider constraints on higher order derivatives, which lead to solutions that are splines of higher degrees.

$$\begin{array}{ll} \text{minimize} & L(f) \\ \text{subject to} & \text{TV}(f^{(k)}) \leq \tau, \\ & f^{(k-j)}(x) = 0, \ \forall x \notin [0,1]. \end{array} \tag{4.24}$$

We consider the family of nonparametric function estimation problems indexed by $0 \leq j \leq k$. This is the analogue of the functional regression problem (4.2) with a total variation constraint on the k-th derivative and a saturation constraint on the $(k-j)$-th derivative. The saturating spline case from the rest the chapter is the special case of (4.24) with $k = 1$, $j = 0$. Widely used cubic natural splines correspond to $k = 3$, $j = 1$. Note that unlike natural splines, which are only defined for some values of j and k, there are no constraints on j and k.

We now show that higher-degree saturating splines solve (4.24) in general. As $f^{(k)}$ is of bounded TV, there exists a measure μ s.t. $f^{(k)}(x) = \int 1(t \leq x)\, d\mu(t)$. Then we have

$$\begin{aligned} f^{(k-j)}(x) &= \int \dots \int f^{(k)}(x)dx \dots dx \\ &= \int \dots \int \int 1(t \le x)\, d\mu(t)dx \dots dx \\ &= j! \int (x-t)_+^j\, d\mu(t) + \sum_{l=0}^{j-1} w_l x^l \end{aligned}$$

for some w_l. In the above, all iterated integrals take place j times.

Note that the constraint that $f^{(k-j)}(x) = 0$ for all $x < 0$ implies that the polynomial term, $\sum_{l=0}^{j-1} w_l x^l$ is identically zero. So, we have

$$f^{(k-j)}(x) = j! \int (x-t)_+^j\, d\mu(t).$$

For $x > 1$, we can remove the nonlinearity, that is, for $x > 1$, $f^{(k-j)}(x)$ is simply the integral of a polynomial in x. We can pull terms involving x out of the integral to get a polynomial in x whose coefficients are nonzero multiples of the first j moments of μ:

$$f^{(k-j)}(x) = j! \sum_{l=0}^{j} \binom{j}{l} x^{j-k} \int (-t)^k\, d\mu(t).$$

Again, we note that as this polynomial is identically zero for infinitely many points, all of the coefficients must be zero. In terms of the measure μ, this means:

$$\int t^l\, d\mu(t) = 0 \quad \text{for } l = 0, \dots, j.$$

This shows that the constraint that the $(k-j)$-th derivative of f saturate translates to constraints on all moments of μ up to the j-th moment.

While the conditional gradient step becomes more complex with the addition of more moment constraints, the approach taken in this chapter can still be applied to (4.24) as long as j is fairly small—the conditional gradient step for (4.24) involves a nonconvex optimization problem over $[0,1]^{j+2}$. This is because we need at least $(j+2)$ point-masses to satisfy the moment constraints. So, fitting quadratic splines that saturate to linear is very easy—in fact the code to do so is essentially identical to that for fitting piecewise linear saturating splines splines—but fitting quadratic splines that saturate to constant is slightly more difficult due to the additional linear constraint on the measure μ. Unfortunately for larger values of j and k, we can no longer hope to find the conditional gradient analytically and must resort to recursive gridding or other global optimization algorithms to find the locations of the new knots.

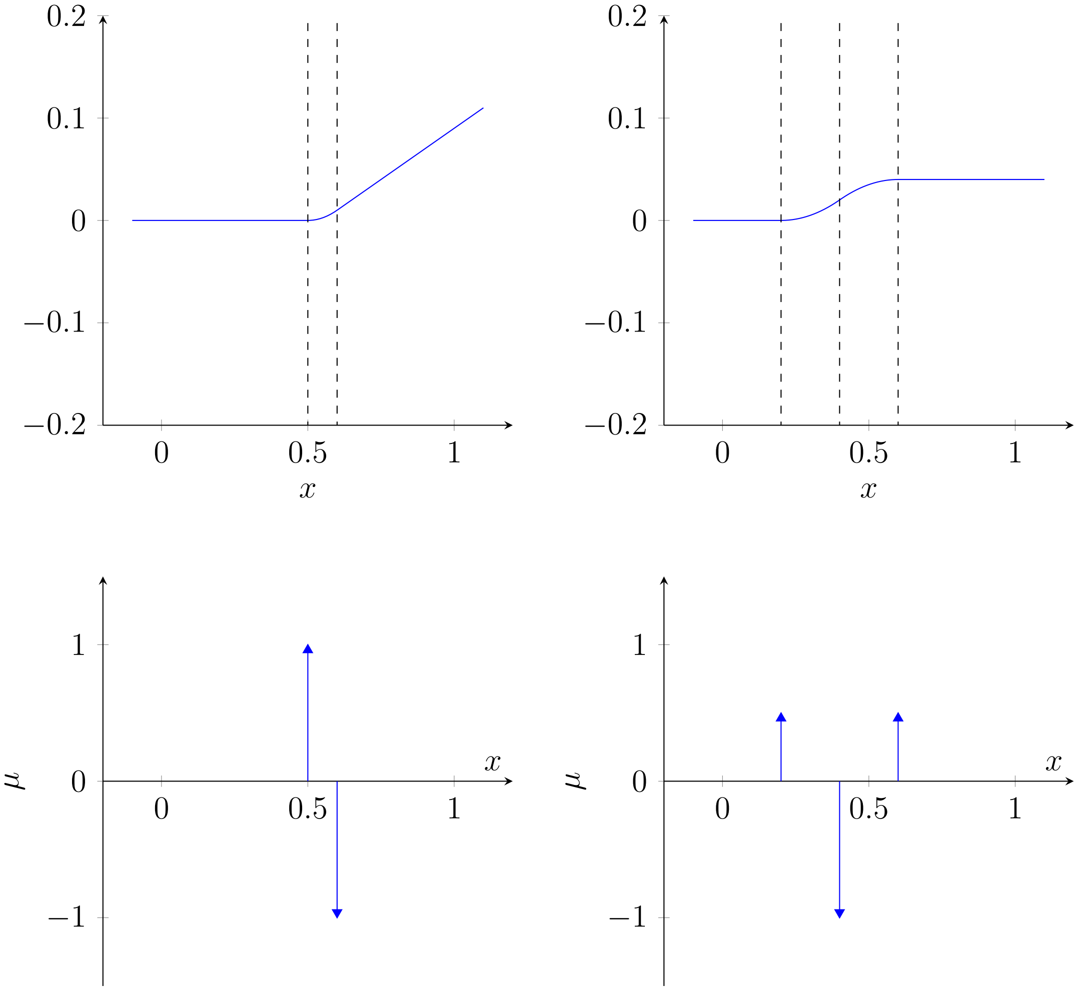


Figure 4.9: The top two plots show conditional gradients for $k = 2$ with $j = 0$ and $j = 1$ respectively. The dashed lines denote the locations of the point masses: when $j = 1$, the conditional gradient consists of three point masses. The bottom plots show the corresponding measures.

4.9 Variations and extensions

While saturation is often a natural prior, the approach we take in this chapter can also be applied to other (convex) variations on (4.9). For example, we could add the constraint that the fitted function is monotone nondecreasing, or takes values in a given interval.

A simple algorithmic extension would be to incorporate nonconvex optimization in the spirit of [14]. At each iteration we adjust the weights of the atomic measure (w), but we could also adjust the knot locations (t). The objective in (4.19) is nonconvex in t_i, but we can still attempt to find a local minimum. As long as we do not increase the objective function the algorithm is still guaranteed to converge [14]. In the case of degree one splines, we can use the fact that the knot points can, without loss of generality, be chosen to be on the data points to make discrete adjustments to the knot locations.

To fit vector-valued functions, for example in multiclass classification, we would need to extend (4.9) to use *vector-valued* measures. This is the natural measure-theoretic analogue to the group-lasso.

In multivariate fitting problems with significant interactions between features generalized additive models may underfit. One possible solution is to use single-layer neural networks: i.e., learn functions of the form

$$x \mapsto \sum_{i=1}^{K} w_i (v_i^T x - t_i)_+.$$

In the above, v_i are constrained to lie in the unit ball. Unfortunately, the conditional gradient step for networks of this form is NP-hard [6]. In many practical applications, however, we might expect that the degree of the interaction is bounded. That is, each v_i has bounded cardinality. If we assume $\|v_i\|_0 \leq 2$, i.e., we only fit pairwise interactions, we can still apply the conditional gradient method. In this case, the fitting function is a sum of functions of pairs of the variables, formed from the basis elements

$$((\cos\theta)x_p + (\sin\theta)x_q - t)_+,$$

with (continuous) parameters θ and t and (index) parameters p and q (i.e., $v = (\cos\theta)e_p + (\sin\theta)e_q$). (This is practical only if d is small enough.) Such functions capture nonlinear relationships between (pairs of) variables.

4.10 Conclusion

In this chapter we propose a modification of the adaptive spline regression model—namely saturation constraints. We show that saturating splines inherit knot-selection from adaptive splines, and have a very important quality in the context of generalized additive models: feature selection. This allows saturating spline generalized additive models to remain interpretable and (crucially) avoid overfitting when applied to multivariate data. We also propose a simple, effective algorithm based on the standard conditional gradient method for solving the saturating spline estimation problem with arbitrary convex losses. Finally, we apply our algorithm to several data sets, demonstrating the simplicity of the resulting models.

Implementation details

We provide a simple, unoptimized implementation in the Rust language. The runtime of our algorithm is dominated by the fully-corrective step, that is, solving the finite-dimensional convex optimization problem (4.19). We solve (4.19) using a proximal Newton method and the standard conditional gradient method with exact linesearch. To be precise, at each iteration, we form the second-order approximation to the objective function

$$f(w) \simeq C + (w - \hat{w})^T \nabla_w f(\hat{w}) + \frac{1}{2}(w - \hat{w})^T \nabla^2 f(\hat{w})(w - \hat{w})$$

which we then minimize (over the constraint set) using the standard conditional gradient method with (exact) linesearch. Note that this is a Newton step with fixed step-length of 1: as in GLMNET [43], we omit a line search in the interest of speed.

We chose to use a proximal Newton method because of its relative simplicity; other standard convex optimization algorithms may give much better practical performance, especially when the number of data points, n, is extremely large.

Proof of theorem 1

Theorem 6. *Fix $x_1, \ldots, x_n \in [0,1]$ and $f : \mathbf{R} \to \mathbf{R}$ with f' (right-continuous) of bounded total variation, and f constant outside of $[0,1]$. Then there exists a degree-one saturating spline $\hat{f}$ that matches f on x_i with $\mathrm{TV}(\hat{f}') \leq \mathrm{TV}(f')$.*

Proof. Without loss of generality, we will assume $f(0) = 0$. Let $\tau = \mathrm{TV}(f')$. As f' has bounded total variation, there exists a measure μ on $[0,1]$ such that $f(x) = f_\mu$:

$$f(x) = \int (x - t)_+ \, d\mu(t).$$

That is, f is a spline with infinitely many knots. The idea is to use Caratheodory's theorem for convex hulls to see that, as we only care about μ in terms of its action on a finite number of functions (basically, we only care about the values of f at x_i), we can replace μ with a measure supported on finitely many points.

To make this idea rigorous, note that the vector

$$v = (f(x_1), \ldots, f(x_n), 0) = \int ((x_1 - t)_+, \ldots, (x_n - t)_+, 1) \, d\mu(t)$$

must lie in convex hull of the (convex) set

$$C = \{\pm(\tau(x_1 - t)_+, \dots, \tau(x_n - t)_+, \tau) : t \in [0,1]\} \subset \mathbf{R}^{n+1}$$

as $\|\mu\|_1 = \tau$. Caratheodory's theorem for convex hulls ensures us that v can be represented as a convex combination of at most $n+2$ points from C. Letting these $n+2$ points be represented by their indicies, $t_1, \dots, t_{n+2}$, and their weights $\alpha_1, \dots, \alpha_{n+2}$ we define $w_j = \alpha_j \tau$ to obtain:

$$f(x_i) = \sum_j w_j (x_i - t_j)_+ = f_\mu(x_i)$$
$$\sum_j w_j = 0.$$

Here $\mu = \sum_j w_j \delta_{t_j}$. As $\sum_j |w_j| = \tau$, we have $\mathrm{TV}(f'_\mu) = \|\mu\|_1 = \tau$.

$\square$

Chapter 5

DeepLoco

In this chapter we revisit the single molecule localization microscopy inverse problem first described in Chapter 2. While ADCG performs well on this problem — in fact, it won the 2016 SMLM 2D high-density localization challenge — we were interested in a completely new way to solve the SMLM inverse problem: posing it as a function approximation problem. Somewhat to our surprise, we show that not only is it possible to directly approximate an inverse function for the SMLM problem but that the resulting function outperforms ADCG: the learned function is orders of magnitude faster than applying ADCG to the maximum-likelihood problem, and it achieves even better accuracy. Once again we see that considering sets of objects as measures allows us to apply (relatively) standard techniques. In this context, however, instead of solving an optimization problem with a measure-valued decision variable, we learn a neural network with a measure as *output*. We show that embedding such measures into a reproducing kernel Hilbert space allows us to apply a natural loss function to train a neural network to output sets of objects.

5.1 Introduction

Visualizing microscopic biological processes is crucial to understanding their function; optical microscopy has been a major tool of biological investigation for over a hundred years. Over the past decade, fundamental physical limits in the resolution of classical microscopy systems have been surmounted by superresolution microscopy techniques [101, 103], enabling the visualization of cellular structures far smaller than before. 3D single-molecule localization microscopy (SMLM) localizes individual fluorophores in 3D space in order to generate an image [59, 32] or facilitate analysis of fluorophore locations. However, these techniques come at a computational cost, requiring advanced algorithms to perform the reconstruction.

While the ultimate goal of most SMLM experiments is to generate high-resolution images, the process involves several intermediate steps with different products. A SMLM experiment proceeds in four stages. First, fluorophores, each a few nanometers in size, are attached to the sample and then stimulated to stochastically fluoresce. Second, a sequence of frames are taken using an optical microscope. Due to the stochastic stimulation, only a small subset of fluorophores are active in each frame. In the third stage, each frame is analyzed to determine the location of each fluorophore active in the frame. (A more sophisticated method processes multiple frames together.) Finally the collection of all locations from all frames is analyzed directly, or used to render high-resolution image in two or three dimensions.

The algorithms we develop in this chapter address the third step of a SMLM experiment, analyzing a single frame to produce a short list of locations of the fluorophores active in the frame. While the aim of most SMLM experiments is to produce an image, we will refer to the task of localizing the fluorophores in a single frame as the SMLM inverse problem.

The localization microscopy community has yet to agree on a universal metric by which to measure performance [102]. Several proposed quality metrics operate on the estimated locations and number of fluorophores directly (such as the Jaccard index at a particular radius), while others apply to the final product of a rendered, high-resolution image (e.g., PSNR). In this chapter we propose, and directly optimize, a new kind of metric: the mean squared error between an infinite-resolution image generated from the estimated fluorophores and the image generated by the ground truth fluorophores. While this metric is an image-based distance, it operates directly on sets of fluorophores.

Our approach to the SMLM inverse problem is to harness the availability of fast, accurate forward-model and noise simulators to train a neural network that maps a single frame to a list of localizations. We do this by attempting to minimize expected loss on simulated data. In the language of statistics, we are attempting to approximate the Bayes' estimator with a neural network. Compared to traditional maximum-likelihood algorithms, our method is easier to calibrate, orders of magnitude faster (once trained), works with a wider variety of noise and forward models, and achieves equal performance on several 2D and 3D datasets.

Our method requires an accurate simulator. Unlike traditional maximum-likelihood/convex optimization based approaches, which make strong assumptions about the forward and noise models (specifically that the log-likelihood is concave and that the forward model is linear), our method can be applied to problems with arbitrary noise statistics, aberrations, and non-linear forward models. Furthermore, we do not require a functional form for the forward simulator, which allows us to handle non-deterministic forward models that take into account aberrations such as dipole effects [32] and, perhaps more importantly, allows us to generate training data directly from a few Z-stacks.

We list three possible disadvantages to our approach. The first is that training a neural network is, at the present time, difficult: unlike convex optimization, there are a plethora of hyperparamters and training usually requires a human in the loop. The second disadvantage is that the method requires an accurate end-to-end generative model that includes the variations and aberrations that will be encountered in the real experimental setup — though arguably this is advantageous: maximum-likelihood approaches are unable to take advantage of this kind of prior knowledge. As we describe in §5.5, the generative model we use is extremely simple and requires only a single Z-stack of experimental data.

Finally, the third disadvantage is common to all applications of neural networks: there is, at present, essentially no theoretical understanding or performance guarantees. For example, training could, in principle, fail for a new experimental setup. Additionally, any given reconstruction could fail. While this is, indeed, a potential shortcoming, experimental evidence suggests that our method is (in practice) at least as reliable as convex methods. Furthermore, all theoretical performance guarantees for convex optimization/maximum likelihood based methods rely on very strong assumptions about the data generation process. We remind the reader that if these assumptions are violated (and they often are, in practice) the conclusions of the theory do not apply. That said, maximum-likelihood based approaches are used extensively in applications where the theoretical assumptions are violated and have a very long history of reliability.

The chapter is organized as follows. First, §5.2 introduces common notation. In §5.3 and §5.4 we describe two bodies of related work, provide background material for our approach, and put our method in context. In §5.5 we give details on our approach, and in §5.6 we describe our experimental setup and results. Finally in §5.7 we describe several possible extensions of our method; some simple, others speculative.

While preparing this manuscript, we discovered another chapter that applies deep learning techniques to STORM microscopy [82]. The major difference between our approaches is that while our algorithm returns a set of localizations, DeepStorm returns a single gridded image. This choice limits the approach to 2D (it's unclear that dense reconstruction could be extended to 3D without, at the very least, a huge increase in computation time), limits rendering to a single scale, and precludes any downstream analysis of the fluorophore locations [25, 84, 31, 73]. Furthermore, the use of an ℓ_1 penalty to encourage sparsity in the reconstructed image introduces an additional parameter that must be tuned and prevents interpretation of the algorithm as an approximation of the Bayes' estimator. With that said, there are some interesting similarities between the approaches: the loss function is essentially a gridded version of our loss function, and both algorithms are substantially faster than existing algorithms. Unfortunately we are unable to directly compare the two approaches as the code for [82] is not yet available.

5.2 Notation and loss functional

One issue with SMLM as an inverse problem is the choice of loss function. In many inverse problems the object to be estimated is an element of a finite-dimensional Hilbert space, in which case the L_2 distance is a natural loss function. SMLM is more complicated: it is unclear how to compare two sets of points. In this section we argue that, in fact, SMLM is not so different from simpler inverse problems: while the intermediate output of a SMLM experiment may be a collection (of varying cardinality) of (possibly weighted) point sources, the final objective of most SMLM experiments is to render an image. This interpretation suggests a natural metric: the squared error of the resulting rendered image. We propose rendering the image at *infinite* resolution for computational efficiency and using the resulting L_2 distance directly as the training loss function.

In this section we introduce some common notation for the remainder of the chapter and formalize the loss function described above. The underlying object to be estimated is a set of points in $\Theta \subset \mathbf{R}^2$ (or $\Theta \subset \mathbf{R}^3$ for 3D SMLM), $\gamma = \{\theta_1, \ldots, \theta_n\}$. Here $\theta_i \in \Theta$ is the location of the i-th fluorophore in space. Note that n, the number of fluorophores active in the frame, is unknown and varies from frame to frame. While the object to be estimated is simply a collection of points, we'll often deal with *weighted* sets of points, of the form $\{(w_1, \theta_1), \ldots, (w_n, \theta_n)\}$ where $w_i \in \mathbf{R}$ and $\theta_i \in \Theta$. The w_i will have different interpretations in different contexts. When we talk about simulating data or using maximum-likelihood estimation, $w_i > 0$ will be the intensity of the i-th active fluorophore in the frame — that is, roughly proportional to how many photons that fluorophore emitted during the exposure. In the output of the neural network, however, w_i will be interpreted as a confidence. We'll often talk about unweighted sets of points (like γ) as weighted collections, in which case we'll take each w_i to be one. Finally, it'll often be convenient to make use of a bijection between weighted collections of (unique) objects in Θ and *finitely-supported* atomic measures on Θ. The bijection associates a weighted collection $\gamma = \{(w_1, \theta_1), \ldots, (w_n, \theta_n)\}$ with the measure

$$\mathcal{M}(\gamma) = \sum_{i=1}^{n} w_i \delta_{\theta_i}.$$

Here δ_θ is a point-mass supported on θ. Similarly, if μ is a finitely-supported atomic measure on Θ, $\mathcal{M}^{-1}(\mu)$ is a well-defined weighted set of points.

Rendering and loss functional

The final stage in many SMLM experiments is to render an image from the localized fluorophores. In practice, localizations must be convolved with a small convolution kernel before they are rendered. This blur serves two purposes: first, it allows an image to be formed, and second it attempts to make explicit uncertainty in the localization, both from the estimation process and from the fact that the fluorophore molecules (which are attached to the molecules of interest) have nonzero spatial extent. While in many SMLM applications the resulting images are rendered on a fine grid, we will simply consider the infinite-resolution image as a functional on $\mathbf{R}^2$ or $\mathbf{R}^3$. Given a convolution kernel $\phi \in L^2(\mathbf{R}^2)$ (or $\mathbf{R}^3$), the image generated by the (confidence-weighted) set $\hat{\gamma}$ is

$$\mathcal{R}_\phi(\{(\hat{w}_1, \hat{\theta}_1), \ldots, (\hat{w}_{\hat{n}}, \hat{\theta}_{\hat{n}})\}) = x \mapsto \sum_i \hat{w}_i \phi(x - \hat{\theta}_i). \tag{5.1}$$

$\mathcal{R}_\phi$ can also be written compactly as a convolution of the measure $\hat{\mu} = \mathcal{M}(\hat{\gamma})$ with the kernel ϕ:

$$\mathcal{R}_\phi(\hat{\mu}) = \phi * \hat{\mu}. \tag{5.2}$$

With this notation in place, we can introduce the family of loss functions we will use. With convolution kernel ϕ, let

$$l_\phi(\gamma, \hat{\gamma}) = \|\mathcal{R}_\phi(\gamma) - \mathcal{R}_\phi(\hat{\gamma})\|_2^2. \tag{5.3}$$

This expands into the following quadratic form in $(w, \hat{w})$:

$$\sum_i \sum_l w_i w_l K(\theta_i, \theta_l) - 2 \sum_i \sum_j w_i \hat{w}_j K(\theta_i, \hat{\theta}_j) + \sum_j \sum_k \hat{w}_j \hat{w}_k K(\hat{\theta}_j, \hat{\theta}_k). \tag{5.4}$$

In the above $K(\theta, \zeta)$ is the positive semi-definite function defined by

$$K(\theta, \zeta) = \langle \phi(\cdot - \theta), \phi(\cdot - \zeta) \rangle_{L^2}. \tag{5.5}$$

If γ is the true collection of fluorophore locations, we take each w_i to be identically one, while a localization algorithm may use $\hat{w}_i$ to encode its confidence that there is a fluorophore at location $\hat{\theta}_i$.

As long as K is known, we can compute (5.3) efficiently, at least when n and $\hat{n}$ are relatively small. If n or $\hat{n}$ are large, any number of truncated or random embeddings will work to approximate (5.3) [93, 110, 26].

For instance, a typical choice of ϕ in applications is the standard Gaussian probability density function at a particular scale σ:

$$\phi_\sigma(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{\|x\|^2}{2\sigma^2}},$$

which corresponds to

$$K(x, y) = \frac{1}{\sqrt{8\pi^2\sigma^2}} e^{-\frac{\|x-y\|_2^2}{4\sigma^2}}. \tag{5.6}$$

As we'll see later in §5.5, more exotic choices are possible. In practice, σ is often chosen to be near the expected localization precision of the system, i.e., 20 to 50 nanometers.

5.3 Maximum-likelihood methods

In this section we briefly describe existing techniques, almost all of which are based on maximum-likelihood estimation. Maximum-likelihood and regularized maximum-likelihood methods for inverse problems have proven to be effective over a wide variety of applications, and SMLM is no exception: the highest-performing SMLM algorithms are all variations on maximum-likelihood estimation [58, 102]. In this section we describe one family of convex approximations to the SMLM maximum-likelihood estimation problem.

These approaches assume additional structure in the measurement process and the noise model, though they seem to work well even when the assumptions aren't met. First, they assume that the measurement process is a function of (only) the positions and intensities of the sources and is given by an operator $\mathcal{I}$. Furthermore, they require that $\mathcal{I}$ is additive in the sources and linear in the intensities:

$$\mathcal{I}\gamma = \sum_i w_i \phi(\theta_i). \tag{5.7}$$

In the above, $\phi : \Theta \to \mathbf{R}^d$ is a known function. In microscopy, ϕ is a spatially-translated and pixelated copy of the microscope's on-axis point-spread function. These algorithms further assume that the negative log-likelihood of the noise distribution is a known convex function $\ell : \mathbf{R}^d \times \mathbf{R}^d \to \mathbf{R}$. For instance, if the per-pixel noise is approximately Gaussian, then $\ell(x, y) = \frac{1}{2}\|x - y\|_2^2$. A maximum-likelihood estimate of γ is a solution to the following optimization problem:

$$\underset{\hat{\gamma}}{\text{minimize}} \;\; \ell(y, \mathcal{I}\hat{\gamma}).$$

Even with these additional assumptions, this optimization problem is quite difficult: $\hat{\gamma}$ is of unknown cardinality, and the objective function is non-convex in the spatial locations of the fluorophores.

One way to avoid these issues is to lift the optimization variable $\hat{\gamma}$ to the measure $\mathcal{M}(\hat{\gamma})$. The additional structure described in (5.7) means that the nonlinear measurement operator $\mathcal{I}$ can be extended to a *linear* operator on measures. For instance, with $\mu = \sum_i w_i \delta_{\theta_i}$:

$$\mathcal{I}\mu = \sum_i w_i \phi(\theta_i) = \int \phi(\theta) d\mu(\theta).$$

This last expression is well-defined for *all* signed measures of finite total variation. As the composition of a linear operator and a convex function is convex, the following optimization problem is convex in the variable μ:

$$\underset{\mu \geq 0}{\text{minimize}} \;\; \ell(y, \mathcal{I}\mu). \tag{5.8}$$

Unfortunately, the solution to (5.8) is, in general, not finitely-supported and thus cannot be interpreted as a weighted collection of points. One heuristic to encourage the solution of (5.8) to be supported on a small number of points is to add a penalty term on the total mass of μ: this is the infinite-dimensional analog of the ℓ_1 norm. This modification results in the following (infinite-dimensional) convex optimization problem:

$$\underset{\mu \geq 0}{\text{minimize}} \;\; \ell(y, \mathcal{I}\mu) + \lambda\mu(\Theta), \tag{5.9}$$

where λ is a positive parameter. It can be shown that the solution to (5.9) is guaranteed to be finitely supported [19, 14], and thus can be interpreted as a weighted collection; in practice, the support of the solution is often extremely sparse. Here $\lambda > 0$ allows us to trade off data fidelity for the cardinality of the support of the estimated measure. State-of-the-art algorithms for SMLM solve (5.9)[14] or a finite-dimensional, gridded analogue of (5.9)[125, 81, 44, 87].

In practice, these algorithms may also require postprocessing: for instance thresholding by removing points for which the estimated intensity $\hat{w}_i$ is low, or by clustering nearby localizations [113]. Of some interest are the myriad of theoretical results concerning (5.9): these results stipulate that if the measurement model $\mathcal{I}$ is accurate and some additional technical assumptions are satisfied, the solution to (5.9) is guaranteed to be close (in some sense) to the ground truth [104, 41].

5.4 Function approximation methods for inverse problems

In this section we briefly discuss related work applying deep learning techniques to inverse problems.

Recently there has been great interest in using deep neural networks for inverse problems, especially problems in imaging [77, 80]. We group the field into into two broad groups: amortized or compiled inference, and iterative, or unrolling approaches.

Amortized or compiled inference attempt to directly learn an approximation $\tilde{F}^{-1}(y)$ (or in some cases an approximation to the full posterior) for the problem $y \simeq F(x)$, often by training a network with a very large number of known (x, y) pairs. In the applications highlighted in [77], including superresolution imaging [66], motion deblurring [122], and denoising [123], the output of $\tilde{F}^{-1}$ is a dense image. Other recent work [65] has attempted to learn an inverse model to classify the hand-written MNIST dataset from a camera system with minimal optics. In this case the predicted output of the network is an integer from zero to nine. Learning function approximators for inverse problems has a rich history, and multi-layer neural networks were used beginning 30 years ago for inverse problems [68], including problems in optics [117]. More contemporary work on *compiled inference* for probabilistic programming [72] also extends this approach.

Unrolling approaches take an existing iterative algorithm and replace some components with a learned operator — either the actual iterative steps themselves (hence the "unrolling") or a proximal operator for algorithms with a proximal step. These approaches can exploit known linearities in the problem. One of the earliest examples is learned iterative shrinkage-thresholding algorithm (LISTA [47]), which unrolled an iterative-shrinkage and thresholding algorithm and learned approximations for the adjoint and gram matrix. ADMM-based approaches [120] learn networks that approximate subproblems in ADMM.

In microscopy, recent work has attempted to learn data-driven methods for upsampling conventionally acquired images [98, 121], although in all cases, these data-driven methods make assumptions about the nature of the system under study. Indeed, the authors of [121] specifically caution against using their method for imaging novel biological structures.

5.5 DeepLoco

In this section we describe how we solve the SMLM inverse problem using a function approximator: we train a neural network to directly minimize expected per-frame loss on simulated data. We first describe the generative model we use to create training data. We then describe the loss function and how it can be extended to arbitrary problems involving weighted collections of objects. Finally we briefly describe the architecture of our network and how we train it.

Data generation

The success of function approximation techniques on novel datasets relies on the availability of vast quantities of labeled training data. We argue that SMLM falls into this class of problems. The combination of a reasonable generative model for fluorophores and a well-understood forward model means that, considered as a machine learning problem, SMLM has essentially *infinite* training data: we can simulate as many training examples as we need. Obviously there is still mismatch between the simulated data and any test data we encounter in the real world; the real question is how this mismatch affects the performance of our algorithms. In this chapter we show that the mismatch is small enough that we can obtain good localizations.

The first step in simulating SMLM data is to generate random collections of fluorophores. For each image we sample the number of fluorophores, n, from a uniform distribution. We then sample n spatial locations independently and uniformly from a 3D box. We sample the fluorophore intensities from a uniform distribution.

Next, we run the collection of fluorophores through a forward model to generate noiseless observations. The forward model we use can be thought of as aggressive data augmentation and is very easy to apply in practice. We use laterally translated — in most experimental setups the PSFs are invariant to translation in X and Y — versions of an empirically measured PSF to generate new data. This approach has several advantages over using a fitted functional form. By taking multiple Z-stacks of different fluorophores (or beads), we can train the network to be robust to aberrations that vary from fluorophore to fluorophore (such as dipole effects [32]). It also removes the critical preprocessing step of fitting a parametric model to the Z-stacks and is hyperparameter-free.

The final step in simulating SMLM data is to add noise to the image. In our experiments we simply use Poisson noise for each pixel. While this is not a great fit for experimental data, we find that the method is robust enough that this mismatch is not an issue. As we discuss in §5.7, including a more accurate noise or background model could improve the performance of our approach.

Loss functions

In §5.2 we introduced a metric for SMLM. While that metric can be used to evaluate the results of a entire SMLM experiment, in this subsection we show how we use it on *single* frames in order to train a neural network. We also describe practical extensions, such as rendering at multiple scales to help training.

To compute the loss on a single frame we set the weights for the true active fluorophores to one, resulting in the target collection $\gamma = \{(1, \theta_1), \ldots, (1, \theta_n)\}$. Note that we do not use weight one to *generate* the image, each fluorophore has a different intensity for the simulation, but we set the weights to one when computing the loss. As the number of fluorophores active in a single image is relatively small (in our experiments at most a few hundred), we use (5.5) to compute l_ϕ. For training we use the Laplacian kernel:

$$K(x,y) = \frac{1}{2\sigma} e^{-\frac{\|x-y\|_1}{\sigma}}, \tag{5.10}$$

which corresponds to using the first modified Bessel function of the second kind as the convolution kernel ϕ.

In our experiments we found that evaluating the loss function at multiple scales during training improves the final performance of the network. The loss function is then

$$\ell(\gamma,\hat{\gamma}) = \sum_i l_{\phi_i}(\gamma,\hat{\gamma}), \tag{5.11}$$

where each ϕ_i is a convolution kernel at a different scale. This loss function can be evaluated using the quadratic form in (5.5) with $K(x,y) = \sum_i K_i(x,y)$, where $K_i(x,y) = \langle \phi_i(\cdot - x), \phi_i(\cdot - y)\rangle_{L^2}$ and each ϕ_i a convolution kernel at scale σ_i.

Generalization to other machine learning problems

Readers familiar with reproducing kernel Hilbert spaces [2] will recognize (5.10) and (5.6) as reproducing kernels. Indeed, another way to think of the metric (5.5) is as the *maximum mean discrepancy* [48] between the measures $\mathcal{M}(\gamma) = \mu$ and $\mathcal{M}(\hat{\gamma}) = \hat{\mu}$. The maximum mean discrepancy between μ and $\hat{\mu}$ is given by

$$\mathrm{MMD}_K(\mu,\hat{\mu}) = \frac{1}{2}\|K\mu - K\hat{\mu}\|^2_{\mathcal{H}}. \tag{5.12}$$

In the above, $\mathcal{H}$ is the RKHS generated by the kernel K. By a slight abuse of notation we use K to denote the linear operator (with codomain $\mathcal{H}$) defined by

$$K\mu = \int K(\theta,\cdot)d\mu(\theta).$$

This suggests an extension of the loss function (5.3) to arbitrary spaces Θ equipped with a kernel K: simply take

$$\ell(\gamma,\hat{\gamma}) = \mathrm{MMD}_K(\mathcal{M}(\gamma),\mathcal{M}(\hat{\gamma})). \tag{5.13}$$

For more on the topic of embeddings of weighted collections of points (and general measures) into RKHS, see [109].

Neural network architecture

We use a fairly standard convolutional neural network architecture. We emphasize that our contributions are the application of neural networks to the localization problem and the loss function described above: the architecture of the neural network we use is essentially arbitrary and almost surely suboptimal. We use the same architecture for both 2D and 3D

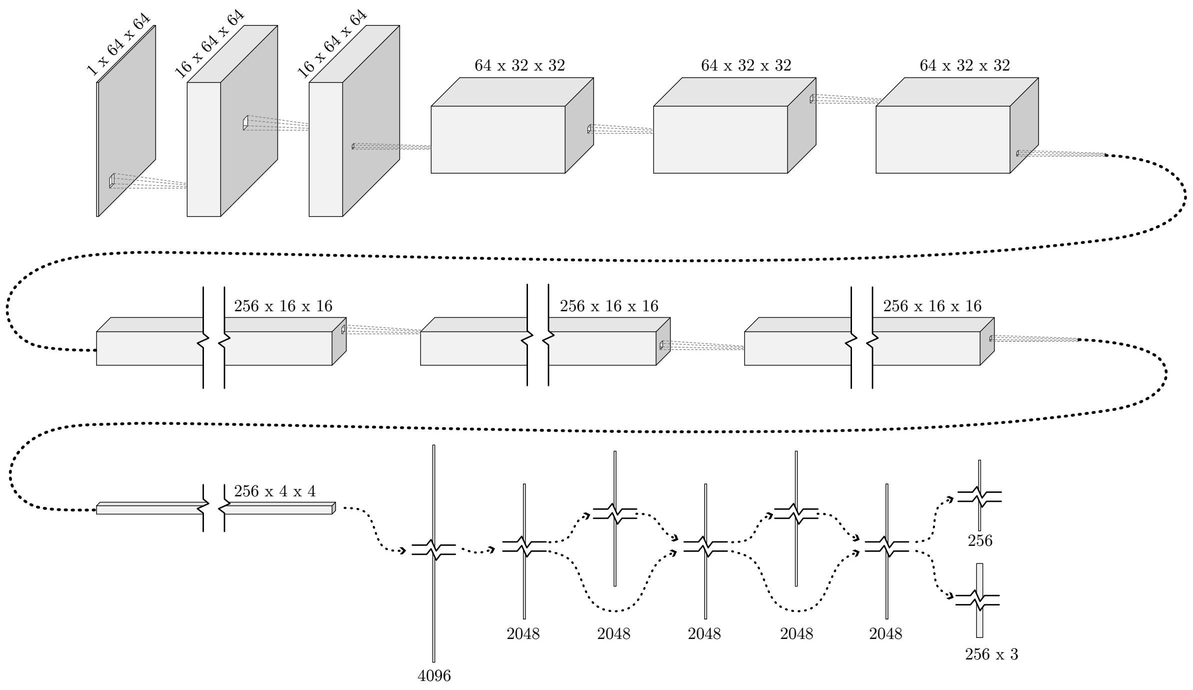


Figure 5.1: A visualization of the DeepLoco architecture. The first two convolutions use 5 x 5 filters, while the remaining convolutions use 3 x 3 filters. Spatial downsampling is by strided convolution; twice using 2 x 2 filters with stride 2 and once with 4 x 4 filters with stride 4.

experiments (except for the final layer, which outputs either two or three spatial coordinates). The first part of the network is fully convolutional (with so-called ReLU nonlinearities) and alternates three times between performing convolutions at given scale and spatial downsampling by strided convolution. This is followed by a two-layer fully-connected ResNet [54]. Finally, the output of the ResNet is fed into two linear layers that output a fixed (but large) number, K, of sources. One linear layer outputs K weights; non-negativity of the weights is enforced by a ReLU nonlinearity. We take K to be much larger than n, the number of true sources; the network is free to set many of the weights to zero. The second linear layer outputs a tensor of size K by 2 (or 3, for 3D localization) that encodes the predicted spatial locations $\hat{\theta}_i$. This layer uses a sigmoid nonlinearity to ensure that the estimated locations remain within a given spatial extent.

Training

We first simulate a batch of data to use as a validation set during training. During each training iteration, we simulate a new batch of training data (both spatial locations and noisy images) and run one step of a stochastic gradient descent variant (in our experiments either SGD with momentum or ADAM). Every few iterations we evaluate the error on the validation set; when the error plateaus we reduce the stepsize and reset the optimization algorithm.

5.6 Experimental results

In this section we compare our algorithm to the existing state-of-the-art algorithms on both simulated and contest data. The SMLM community has established several [102] contests to objectively benchmark the performance of these algorithms for both 2D and 3D localization. In both cases, we compare to the best-performing algorithms for each task: the Alternating Descent Conditional Gradient method (ADCG) [14] which won the 2016 high-density 2D challenge, and Spliner [5] (also referred to as CSpline), the winner (for the astigmatic PSF in low density and the double-helix PSF in both high and low densities) of 2016 3D challenge. Note that for data generated from our simulator we give both competing algorithms a handicap : we run them across a range of parameter settings and post-hoc pick the one that gives the highest Jaccard index. While not feasible in the real world, this helps lessen the risk of “parameter-hacking”.

To compare the different algorithms in different regimes we vary both the source density (number of simultaneously active sources per frame) and the signal-to-noise ratio of the point sources. Reconstruction accuracy can be measured by comparing the estimated locations directly to the true locations or by comparing high (or infinite) resolution rendered images; we compute metrics of both types. In all experiments that compare localizations directly we use simple postprocessing on the output of our method: we cluster the output points into connected components in a thresholded distance graph and then remove points with low confidences. To compare localizations directly we follow [102] by first solving an assignment problem in euclidean space: matching each detected point to a nearby true source point in a manner that minimizes the total euclidean distance between pairs of points. We then consider all pairs of matched points closer than a threshold (in our experiments, either 50nm or 100nm) as true positives (TP), and all other source points as false positives (FP). Similarly, missed ground truth points are considered false negatives (FN). We then compute the Jaccard index, J,

$$\mathrm{J} = \frac{\mathrm{TP}}{\mathrm{FN} + \mathrm{FP} + \mathrm{TP}}. \tag{5.14}$$

A Jaccard index of 1.0 indicates a perfect matching — all source points are recovered within the tolerance radius, with no spurious detections. With a fixed point matching we also compute the mean distance between matched pairs, in either x (2D) or x and z (3D).

Finally, we report the image loss defined in (5.3) for a Gaussian convolution kernel with $\sigma = 50$nm.

2D SMLM

We first compare ADCG and DeepLoco on synthetic data to investigate how each algorithm performs under varying source conditions. The two dimensional synthetic data is generated from the 2016 SMLM 2D contest Z-stack. Fluorophores and noisy images are generated within 350nm of the focal plane over a $6.4\mu m \times 6.4\mu m$ area with a per-pixel resolution of 100nm. We present the results in Figure 5.2. DeepLoco and ADCG have similar recognition accuracy (Jaccard index) and spatial localization accuracy across a variety of point densities and source intensities.

We also compare ADCG and DeepLoco on the MT0.N1.LD and MT0.N1.HD datasets from the 2016 SMLM contest in Table 5.2. We find that DeepLoco comparably if not slightly better than ADCG in the low and high-density cases. This is likely due to the relatively simplistic Gaussian PSF model used by ADCG, which is a poor fit for some of the datasets' point sources which do not lie close to the focal plane.

3D SMLM

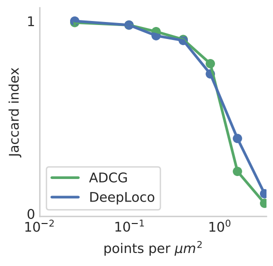
Three-dimensional SMLM uses point spread functions that vary with source depth, allowing a fluorophore's position to be estimated in all three coordinates from a single frame. We compare DeepLoco to Spliner with two different PSFs: an astigmatic PSF [59] and a double-helix PSF [88].

We first compare the algorithms on synthetic data generated from the SMLM challenge calibration Z-stacks. These experiments (in Figure 5.3) show that DeepLoco significantly outperforms Spliner in terms of Jaccard index, while Spliner is slightly better in localization accuracy with the astigmatic PSF.

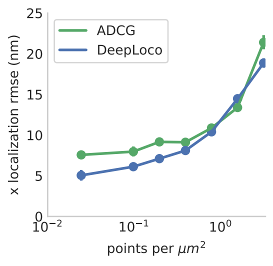
We next compare the algorithms on the MT0.N1.LD and MT0.N1.HD datasets from the 2016 SMLM challenge. We evaluate the results visually in Figure 5.4 and using quantitative metrics in Table 5.1. DeepLoco significantly outperforms Spliner in terms of Jaccard index and the kernel loss. The two algorithms are comparable in terms of spatial localization accuracy, except for the low-density double-helix data, where Spliner significantly outperforms DeepLoco.

Runtime

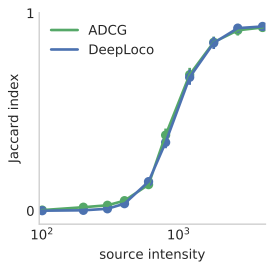
DeepLoco is significantly faster than ADCG and Spliner. While DeepLoco runs in (essentially) constant time regardless of fluorophore density, both ADCG and Spliner have iterative components that scale with the number of input sources. We run the algorithms on subsets of data from the SMLM 2016 contest to capture the runtime dependence on source density (Table 5.3). DeepLoco was run on an Amazon Web Services `p3.2xlarge` instance with a



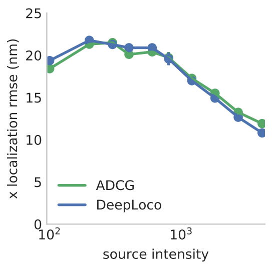
(a)



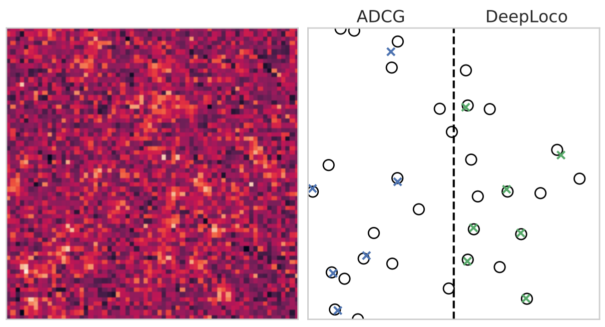
(b)



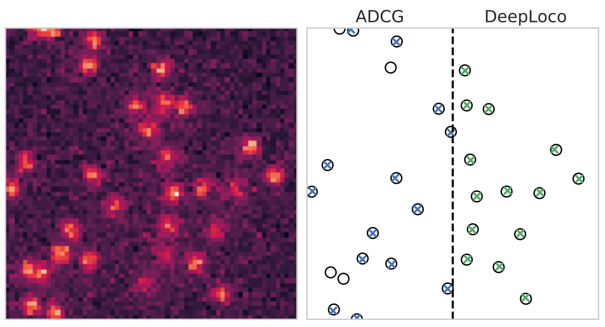
(c)



(d)



(e)



(f)

Figure 5.2: 2D localization using DeepLoco and ADCG. All point matchings are computed with a 50nm tolerance radius. We vary the source density (**a.** and **b.**) and source intensity (**c.** and **d.**) and compare localization performance via Jaccard index and RMSE in the x-coordinate. For experiments with varying source intensity we use a fixed density of 0.2 molecules per μm^2. We find DeepLoco performs virtually identically to ADCG across the full range of tested parameters. **e.** and **f.** show example source images and the localizations returned by each algorithm.

single Nvidia V100 GPU. ADCG and Spliner were run on the equivalent of an Amazon Web Services `c4.8xlarge` machine with 18 physical cores (though ADCG and Spline are not optimized to take advantage of multiple cores).

On high-density data, DeepLoco is roughly 40000 times faster than ADCG and 2000 times faster than Spliner.

5.7 Extensions and variations

In this section we briefly describe some extensions to this work.

The successful application of machine learning techniques to the SMLM inverse problem opens up a new path to better SMLM algorithms: developing more accurate simulators. Raw SMLM data rarely looks like the output of the simulators we use in this chapter. This discrepancy has two main causes: structured background fluorescence and aberrations in

Low Density SMLM 2016 3D Data

PSF	Algorithm	Jaccard	x RMSE (nm)	z RMSE (nm)	kernel loss
3D-AS	DeepLoco	0.83 ± 0.33	11.83 ± 11.02	24.80 ± 17.85	0.38 ± 0.54
	Spliner	0.77 ± 0.37	9.88 ± 9.77	22.34 ± 18.45	0.49 ± 0.75
3D-DH	DeepLoco	0.84 ± 0.30	17.25 ± 13.52	22.35 ± 16.71	0.67 ± 0.88
	Spliner	0.74 ± 0.36	11.85 ± 12.04	15.99 ± 15.04	0.65 ± 0.94

High Density SMLM 2016 3D Data

PSF	Algorithm	Jaccard	x RMSE (nm)	z RMSE (nm)	kernel loss
3D-AS	DeepLoco	0.51 ± 0.16	16.12 ± 6.56	27.30 ± 8.56	6.49 ± 3.41
	Spliner	0.37 ± 0.16	14.89 ± 7.42	27.83 ± 10.91	9.18 ± 4.56
3D-DH	DeepLoco	0.39 ± 0.19	23.60 ± 9.32	28.89 ± 9.98	9.43 ± 6.35
	Spliner	0.27 ± 0.14	24.26 ± 11.87	25.13 ± 11.93	11.36 ± 5.13

Table 5.1: Reconstruction metrics for various algorithms across 3D datasets. All point matchings are computed with a 100nm tolerance radius. Values are mean ± standard deviation.

Density	Algorithm	Jaccard	x RMSE (nm)	kernel loss
Low	DeepLoco	0.89 ± 0.26	11.61 ± 12.27	0.22 ± 0.45
	ADCG	0.79 ± 0.35	14.95 ± 13.27	0.50 ± 0.84
High	DeepLoco	0.61 ± 0.16	18.22 ± 7.24	5.22 ± 3.29
	ADCG	0.53 ± 0.15	19.36 ± 7.43	6.63 ± 3.60

Table 5.2: Reconstruction metrics for DeepLoco and ADCG across high and low-density 2D datasets. All point matchings are computed with a 100nm tolerance radius.Values are mean ± standard deviation.

Table 5.3: Algorithm runtime for different datasets from the 2016 SMLM challenge

		Algorithm (frames/second)		
PSF	Density	ADCG	Spliner	DeepLoco
2D	low	2.2		20859
	high	0.44		20859
3D-AS	low		40.6	20859
	high		9.9	19742
3D-DH	low		63.5	20859
	high		9.3	20806

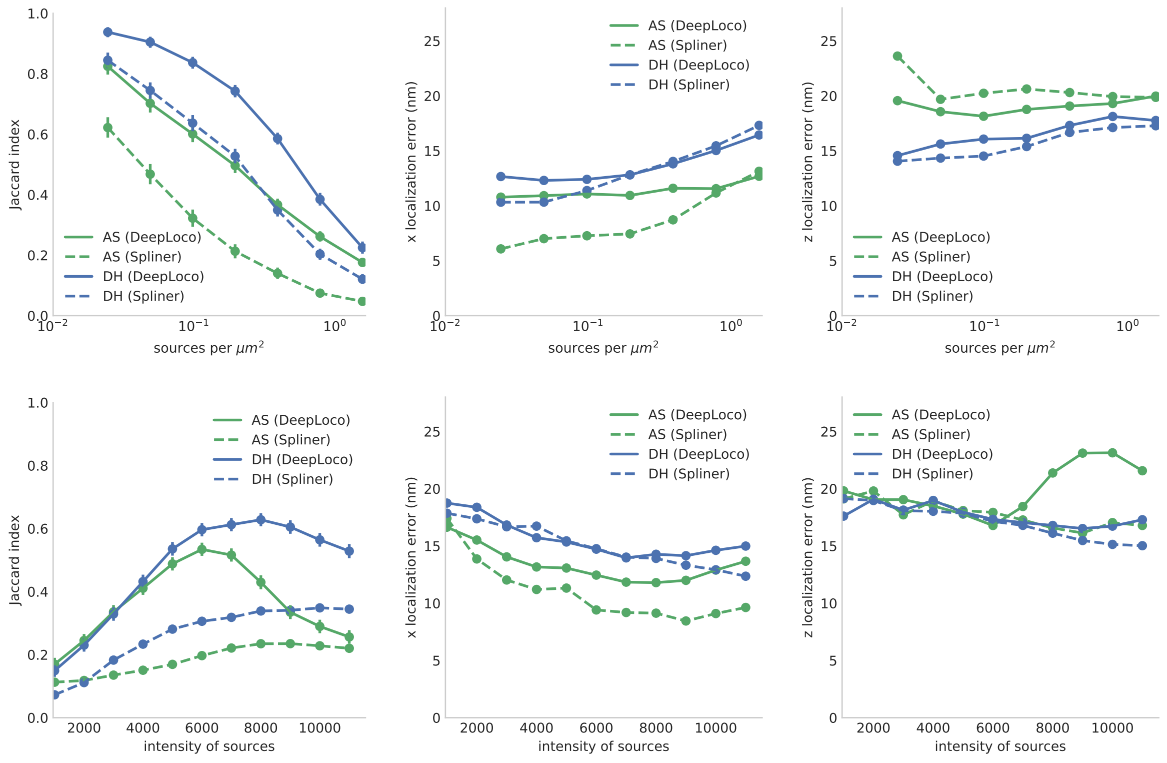


Figure 5.3: 3D localization of synthetic data by DeepLoco and Spliner. All point matchings are computed with a 50nm tolerance radius. For experiments with varying source intensity we use a fixed density of 0.2 molecules per μm^2. Error bars reflect a 95% confidence interval on the mean estimated by the bootstrap. While the Jaccard index suggests that localization at very high densities (i.e. 10 sources per μm) fails, both Spliner and DeepLoco produce reasonable images — suggesting that if the goal is to render an image the Jaccard index is not a representative metric.

the optical system. In existing algorithms these issues are typically handled using delicate, heuristic preprocessing steps. Our approach suggests a different approach: training a neural network with a simulator that directly models background noise and aberrations.

One issue exposed in our experiments is the sensitivity of our networks to mismatch between the training distribution and the test distribution. For instance, and unlike any existing approach, our network can return *less* accurate localizations for extremely bright fluorophores. This bizarre behavior can be explained by a mismatch between the training distribution and the test distribution: networks trained to localize (relatively) low SNR sources do not generalize to higher SNR sources. Investigating and mitigating this issue would increase confidence in our approach.

From a statistical point of view, processing each frame independently is highly suboptimal: the photophysics of the fluorophore molecules is such that fluorophores active in one

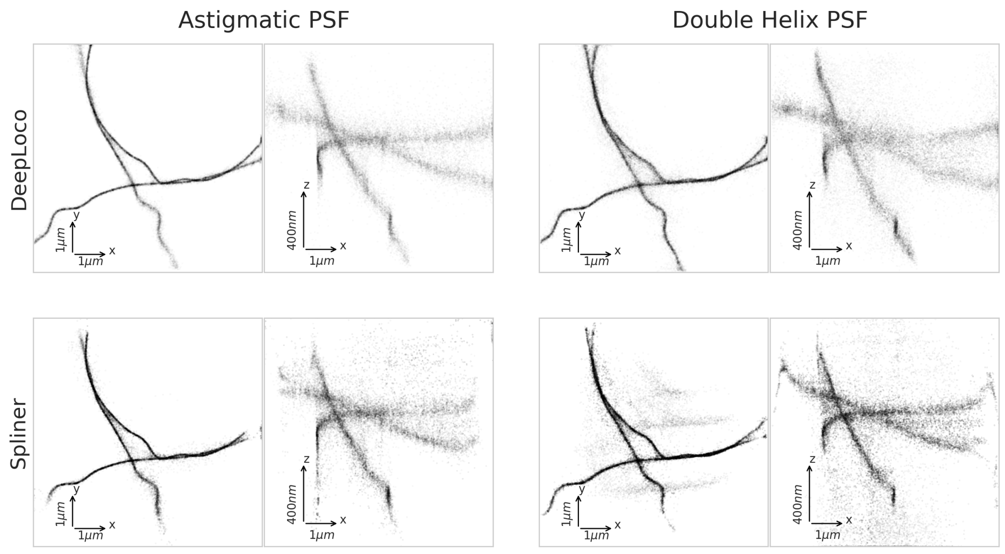


Figure 5.4: Super-resolution images rendered by DeepLoco and Spliner on the SMLM2016 MT0.N1.HD high-density dataset for both astigmatic and double-helix PSFs.

frame are often active in neighboring frames. Because localization accuracy is limited by the photon count of each emitter [85], analyzing multiple frames together could greatly increase accuracy. It's also possible that analyzing multiple frames could help localizing much higher densities of emitters: at high densities emitters that are spatially overlapping still blink independently. Several existing techniques analyze multiple frames [27, 18, 111] but are limited by computational cost. The increased processing speed provided by using a feedforward neural network might allow efficient analysis of multiple frames and would be straightforward to implement.

Programmable spatial light modulators allow experimenters to change the point spread function of the microscope to, for instance, localize over much larger depths [107]. A neural network that takes the phase mask in addition to the raw image as input might be able to localize using a huge variety of point-spread functions.

A related question is how to reduce the expense of training the neural network for new experimental conditions (e.g. different noise levels or background conditions). While training time in our experiments was typically on the order of a few hours, preliminary experiments with warmstarting the optimization from pre-trained networks suggest this can be significantly reduced.

The loss function we introduce is a natural loss function for a variety of classification tasks where the labels are collections of parametric objects, for instance object detection [39]. Comparing the maximum mean discrepancy to existing loss functions would be interesting.

Another possibility for SMLM or other classification problems with set-valued labels is to use different distances between discrete measures: for instance unbalanced optimal transport [23].

Finally, it is unclear how other machine learning techniques would do if applied to this problem. Indeed, it is quite possible that the machinery of deep learning is unnecessary and a simpler algorithm such as k-nearest neighbors would suffice.

5.8 Conclusion

A novel kernel-based loss function allowed us to train a neural network to directly localize sparse emitters in both two and three dimensions. DeepLoco is orders of magnitude faster than existing approaches, while achieving comparable accuracy. The success of DeepLoco suggests that there are regimes where coupling a naive black-box simulator with machine learning can efficiently solve inverse problems; even problems with complex, structured outputs. More physically accurate simulation, including accurately modeling phenomena that would preclude the application of traditional optimization-based approaches, could further improve accuracy.

Bibliography

[1] A. Agarwal, A. Anandkumar, P. Jain, and P. Netrapalli. "Learning sparsely used overcomplete dictionaries via alternating minimization". In: *arXiv preprint arXiv:1310.7991* (2013).

[2] N. Aronszajn. "Theory of reproducing kernels". In: *Transactions of the American mathematical society* 68.3 (1950), pp. 337–404.

[3] S. Arora, R. Ge, T. Ma, and A. Moitra. "Simple, efficient, and neural algorithms for sparse coding". In: *arXiv preprint arXiv:1503.00778* (2015).

[4] D. Arthur and S. Vassilvitskii. "k-means++: The advantages of careful seeding". In: *Proceedings of the eighteenth annual ACM-SIAM symposium on discrete algorithms.* 2007, pp. 1027–1035.

[5] H. P. Babcock and X. Zhuang. "Analyzing Single Molecule Localization Microscopy Data Using Cubic Splines". In: *Scientific Reports* 7.1 (2017), pp. 1–8. eprint: 083402 (10.1101).

[6] F. Bach. "Breaking the Curse of Dimensionality with Convex Neural Networks". In: *Journal of Machine Learning Research* 18.19 (2017), pp. 1–53.

[7] F. Bach. "Convex relaxations of structured matrix factorizations". In: *arXiv preprint arXiv:1309.3117* (2013).

[8] F. Bach, S. Lacoste-Julien, and G. Obozinski. "On the equivalence between herding and conditional gradient algorithms". In: *ICML* (2012).

[9] W. Bajwa, J. Haupt, A. Sayeed, and R. Nowak. "Compressed channel sensing: A new approach to estimating sparse multipath channels". In: *Proc. IEEE* 98.6 (2010), pp. 1058–1076.

[10] R. Baraniuk and P. Steeghs. "Compressive radar imaging". In: *In IEEE Radar Conf., Waltham, MA* (2007), pp. 128–133.

[11] D. M. B.L.R. *DaISy: Database for the Identification of Systems,*

[12] N. Boyd, T. Hastie, S. Boyd, B. Recht, and M. Jordan. "Saturating Splines and Feature Selection". In: *Journal of Machine Learning Research* 17.178 (2017).

[13] N. Boyd, E. Jonas, H. Babcock, and B. Recht. "DeepLoco: Fast 3D Localization Microscopy Using Neural Networks". In: *bioRxiv* (2018).

[14] N. Boyd, G. Schiebinger, and B. Recht. “The Alternating Descent Conditional Gradient Method for Sparse Inverse Problems”. In: *SIAM Journal on Optimization* 27.2 (2017), pp. 616–639.

[15] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein. “Distributed Optimization and Statistical Learning via the Alternating Direction Method of Multipliers”. In: *Foundations and Trends in Machine Learning* 3.1 (2011), pp. 1–122.

[16] K. Bredies and H. K. Pikkarainen. “Inverse problems in spaces of measures”. In: *ESAIM: Control, Optimisation and Calculus of Variations* 19 (01 Jan. 2013), pp. 190–218.

[17] S. Burer and R. D. Monteiro. “A nonlinear programming algorithm for solving semidefinite programs via low-rank factorization”. English. In: *Mathematical Programming* 95.2 (2003), pp. 329–357.

[18] D. T. Burnette, P. Sengupta, Y. Dai, J. Lippincott-Schwartz, and B. Kachar. “Bleaching/blinking assisted localization microscopy for superresolution imaging using standard fluorescent molecules”. In: *Proceedings of the National Academy of Sciences* 108.52 (2011), pp. 21081–21086.

[19] E. Candes and C. Fernandez-Granda. “Towards a mathematical theory of super resolution”. In: *Comm. Pure Appl. Math* (2013).

[20] E. Candes and B. Recht. “Exact matrix completion via convex optimization”. In: *Communications of the ACM* 55.6 (2012), pp. 111–119.

[21] V. Chandrasekaran, B. Recht, P. A. Parrilo, and A. S. Willsky. “The Convex Geometry of Linear Inverse Problems.” In: *Foundations of Computational Mathematics* 12.6 (2012), pp. 805–849.

[22] P. Charbonnier, L. Blanc-Feraud, G. Aubert, and M. Barlaud. “Deterministic Edge-preserving Regularization in Computed Imaging”. In: *Trans. Img. Proc.* 6.2 (Feb. 1997), pp. 298–311.

[23] L. Chizat, G. Peyré, B. Schmitzer, and F.-X. Vialard. “Scaling algorithms for unbalanced transport problems”. In: *arXiv preprint arXiv:1607.05816* (2016).

[24] A. Chouldechova and T. Hastie. “Generalized additive model selection”. In: *ArXiv e-prints* (June 2015). arXiv: 1506.03850 [stat.ML].

[25] C. Coltharp, X. Yang, and J. Xiao. “Quantitative analysis of single-molecule superresolution images”. In: *Current opinion in structural biology* 28 (2014), pp. 112–121.

[26] A. Cotter, J. Keshet, and N. Srebro. “Explicit approximations of the Gaussian kernel”. In: *arXiv preprint arXiv:1109.4603* (2011).

[27] S. Cox et al. “Bayesian localization microscopy reveals nanoscale podosome dynamics”. In: *Nature methods* 9.2 (2012), p. 195.

[28] M. C. D. Malioutov and A. Willsky. “A sparse signal reconstruction perspective for source localization with sensor arrays”. In: *IEEE Trans. Signal Process* 53.8 (2005), pp. 3010–3022.

[29] C. De Boor. *A Practical Guide to Splines*. Applied mathematical sciences. Berlin: Springer, 2001.

[30] V. Demyanov and A. Rubinov. *Approximate methods in optimization problems*. Modern analytic and computational methods in science and mathematics. American Elsevier Pub. Co., 1970.

[31] H. Deschout, A. Shivanandan, P. Annibale, M. Scarselli, and A. Radenovic. “Progress in quantitative single-molecule localization microscopy”. In: *Histochemistry and cell biology* 142.1 (2014), pp. 5–17.

[32] A. von Diezmann, Y. Shechtman, and W. Moerner. “Three-Dimensional Localization of Single Molecules for Super-Resolution Imaging and Single-Particle Tracking”. In: *Chemical Reviews* (2017).

[33] M. Duarte and R. Baraniuk. “Spectral compressive sensing”. In: *Applied and Computational Harmonic Analysis* 35.1 (2013), pp. 111–129.

[34] N. Dunford, J. T. Schwartz, W. G. Bade, and R. G. Bartle. *Linear operators, Part 1*. Wiley-interscience New York, 1958.

[35] J. Dunn and S. Harshbarger. “Conditional gradient algorithms with open loop step size rules”. In: *Journal of Mathematical Analysis and Applications* 62.2 (1978), pp. 432–444.

[36] B. Efron and T. Hastie. *Computer Age Statistical Inference*. Institute of Mathematical Statistics Monographs. Cambridge University Press, 2016.

[37] C. Ekanadham, D. Tranchina, and E. P. Simoncelli. “Neural spike identifcation with continuous basis pursuit”. In: *Computational and Systems Neuroscience (CoSyNe), Salt Lake City, Utah* (2011).

[38] D. Evanko. “Primer: fluorescence imaging under the diffraction limit”. In: *Nature Methods* 6 (2009), pp. 19–20.

[39] M. Everingham et al. “The Pascal Visual Object Classes Challenge: A Retrospective”. In: *International Journal of Computer Vision* 111.1 (Jan. 2015), pp. 98–136.

[40] A. Fannjiang, T. Strohmer, and P. Yan. “Compressed remote sensing of sparse objects”. In: *SIAM J. Imag. Sci.* 3.3 (2010), pp. 595–618.

[41] C. Fernandez-Granda. “Super-resolution of point sources via convex programming”. In: *Information and Inference: A Journal of the IMA* 5.3 (2016), pp. 251–303.

[42] G. B. Folland. *Real Analysis*. 1999.

[43] J. Friedman, T. Hastie, and R. Tibshirani. “Regularization Paths for Generalized Linear Models via Coordinate Descent”. In: *Journal of Statistical Software* 33.1 (2010), pp. 1–22.

[44] S. Gazagnes, E. Soubies, and L. Blanc-Féraud. "High density molecule localization for super-resolution microscopy using CEL0 based sparse approximation". In: *Biomedical Imaging (ISBI 2017), 2017 IEEE 14th International Symposium on*. IEEE. 2017, pp. 28–31.

[45] J. Giesen, M. Jaggi, and S. Laue. "Approximating Parameterized Convex Optimization Problems". In: *ACM Trans. Algorithms* 9.1 (Dec. 2012), 10:1–10:17.

[46] P. J. Green and B. W. Silverman. *Nonparametric Regression and Generalized Linear Models: a Roughness Penalty Approach*. Monographs on statistics and applied probability. Boca Raton, London, New York: Chapman & Hall, 1994.

[47] K. Gregor and Y. Lecun. "Learning Fast Approximations of Sparse Coding". In: *ICML 2010* 152.3 (2005), pp. 318–326. eprint: arXiv:1105.5307v1.

[48] A. Gretton, K. M. Borgwardt, M. J. Rasch, B. Schölkopf, and A. Smola. "A kernel two-sample test". In: *Journal of Machine Learning Research* 13.Mar (2012), pp. 723–773.

[49] B. I. Group. *Benchmarking of Single-Molecule Localization Microscopy Software*. 2013.

[50] A. Hansson, Z. Liu, and L. Vandenberghe. "Subspace System Identification via Weighted Nuclear Norm Optimization". In: *CoRR* abs/1207.0023 (2012).

[51] Z. Harchaoui, A. Juditsky, and A. Nemirovski. "Conditional gradient algorithms for norm-regularized smooth convex optimization". In: *Mathematical Programming* (2014), pp. 1–38.

[52] T. Hastie and R. Tibshirani. *Generalized Additive Models*. Vol. 43. CRC Press, 1990.

[53] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning*. Springer Series in Statistics. New York, NY, USA: Springer New York Inc., 2001.

[54] K. He, X. Zhang, S. Ren, and J. Sun. "Deep residual learning for image recognition". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.

[55] M. Herman and T. Strohmer. "High-resolution radar via compressed sensing". In: *IEEE Trans. Signal Process.* 57.6 (2009), pp. 2275–2284.

[56] R. Hettich and K. O. Kortanek. "Semi-infinite programming: theory, methods, and applications". In: *SIAM review* 35.3 (1993), pp. 380–429.

[57] H. Hindi. "A tutorial on optimization methods for cancer radiation treatment planning". In: *American Control Conference (ACC), 2013*. June 2013, pp. 6804–6816.

[58] S. Holden and D. Sage. "Imaging: super-resolution fight club". In: *Nature Photonics* 10.3 (2016), pp. 152–153.

[59] B. Huang, W. Wang, M. Bates, and X. Zhuang. "Three-dimensional super-resolution imaging by stochastic optical reconstruction microscopy". In: *Science* 319.5864 (2008), pp. 810–813.

[60] M. Jaggi. “Revisiting Frank-Wolfe: Projection-Free Sparse Convex Optimization”. In: *ICML* (2013).

[61] M. Jaggi, M. Sulovsk, et al. “A simple algorithm for nuclear norm regularized problems”. In: *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*. 2010, pp. 471–478.

[62] P. Jain, P. Netrapalli, and S. Sanghavi. “Low-rank matrix completion using alternating minimization”. In: *Proceedings of the forty-fifth annual ACM symposium on Theory of computing*. ACM. 2013, pp. 665–674.

[63] S. G. Johnson. *The NLopt nonlinear-optimization package*. 2011.

[64] R. H. Keshavan. “Efficient algorithms for collaborative filtering”. PhD thesis. Stanford University, 2012.

[65] G. Kim, S. Kapetanovic, R. Palmer, and R. Menon. “Lensless-camera based machine learning for image classification”. In: *arXiv* (Sept. 2017), pp. 11–13. eprint: 1709.00408.

[66] J. Kim, J. K. Lee, and K. M. Lee. “Accurate Image Super-Resolution Using Very Deep Convolutional Networks”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 38.2 (Nov. 2015), pp. 295–307. arXiv: 1511.04587.

[67] S. Kim, K. Koh, S. Boyd, and D. Gorinevsky. “ℓ_1 Trend Filtering”. In: *SIAM review* 51.2 (2009), pp. 339–360.

[68] Kitamura and Qing. “Neural network application to solve Fredholm integral equations of the first kind”. In: *International Joint Conference on Neural Networks*. IEEE, 1989, 589 vol.2.

[69] D. Koller and N. Friedman. *Probabilistic Graphical Models: Principles and Techniques - Adaptive Computation and Machine Learning*. The MIT Press, 2009.

[70] S. Lacoste-Julien, F. Lindsten, and F. Bach. “Sequential Kernel Herding: Frank-Wolfe Optimization for Particle Filtering”. In: *AISTATS* (2015).

[71] S. Laue. “A Hybrid Algorithm for Convex Semidefinite Optimization”. In: *Proceedings of the 29th International Conference on Machine Learning (ICML-12)*. Ed. by J. Langford and J. Pineau. ICML ’12. Edinburgh, Scotland, GB: Omnipress, July 2012, pp. 177–184.

[72] T. A. Le, A. G. Baydin, and F. Wood. “Inference Compilation and Universal Probabilistic Programming”. In: *arXiv* (2016). eprint: 1610.09900.

[73] F. Levet et al. “SR-Tesseler: a method to segment and quantify localization-based super-resolution microscopy data”. In: *Nature methods* 12.11 (2015), p. 1065.

[74] M. Lichman. *UCI Machine Learning Repository*. 2013.

[75] Y. Lin and H. Zhang. “Component selection and smoothing in multivariate nonparametric regression”. In: *Ann. Statist.* 34.5 (Oct. 2006), pp. 2272–2297.

[76] H.-Y. Liu et al. “3D imaging in volumetric scattering media using phase-space measurements”. In: *Opt. Express* 23.11 (June 2015), pp. 14461–14471.

[77] A. Lucas, M. Iliadis, R. Molina, and A. K. Katsaggelos. “Using Deep Neural Networks for Inverse Problems in Imaging”. In: *IEEE SIgnal ProcESSIng MagazInE* 1053.5888/18 (2018).

[78] D. Malioutov, M. Çetin, and A. S. Willsky. “A sparse signal reconstruction perspective for source localization with sensor arrays”. In: *Signal Processing, IEEE Transactions on* 53.8 (2005), pp. 3010–3022.

[79] E. Mammen and S. van de Geer. “Locally adaptive regression splines”. In: *Ann. Statist.* 25.1 (Feb. 1997), pp. 387–413.

[80] M. T. McCann, K. H. Jin, and M. Unser. “A review of convolutional neural networks for inverse problems in imaging”. In: *arXiv* (2017).

[81] J. Min et al. “FALCON: Fast and unbiased reconstruction of high-density super-resolution microscopy data”. In: *Scientific Reports* 4 (2014), pp. 1–9.

[82] E. Nehme, L. E. Weiss, T. Michaeli, and Y. Shechtman. “Deep-STORM: Super Resolution Single Molecule Microscopy by Deep Learning”. In: *arXiv preprint arXiv:1801.09631* (2018).

[83] P. Netrapalli, P. Jain, and S. Sanghavi. “Phase retrieval using alternating minimization”. In: *Advances in Neural Information Processing Systems*. 2013, pp. 2796–2804.

[84] P. R. Nicovich, D. M. Owen, and K. Gaus. “Turning single-molecule localization microscopy into a quantitative bioanalytical tool”. In: *Nature protocols* 12.3 (2017), p. 453.

[85] R. J. Ober, A. Tahmasbi, S. Ram, Z. Lin, and E. S. Ward. “Quantitative Aspects of Single-Molecule Microscopy: Information-theoretic analysis of single-molecule data”. In: *IEEE signal processing magazine* 32.1 (2015), pp. 58–69.

[86] R. Ostrovsky, Y. Rabani, L. J.Schulman, and C. Swamy. “The Effectiveness of *L*loyd-Type Methods for the *k*-Means Problem”. In: *Journal of the ACM* 59.6 (2012), 28:1–28:22.

[87] M. Ovesný, P. Kfffdfffdížek, Z. Švindrych, and G. M. Hagen. “High density 3D localization microscopy using sparse support recovery.” In: *Optics express* 22.25 (2014), pp. 31263–76.

[88] S. R. P. Pavani et al. “Three-dimensional, single-molecule fluorescence imaging beyond the diffraction limit by using a double-helix point spread function.” In: *Proceedings of the National Academy of Sciences of the United States of America* 106.9 (2009), pp. 2995–2999.

[89] A. Petersen, D. Witten, and N. Simon. “Fused Lasso Additive Model”. In: *Journal of Computational and Graphical Statistics* (2015), pp. 1–37.

[90] B. G. R. de Prony. "Essai experimental et analytique: sur les lois de la dilatabilite de fluides elastique et sur celles de la force expansive de la vapeur de l'alkool, a differentes temperatures". In: *Journal de l'ecole Polytechnique* 1.22 (1795), pp. 24–76.

[91] F. Pukelsheim. *Optimal design of experiments*. Vol. 50. siam, 1993.

[92] K. G. Puschmann and F. Kneer. "On super-resolution in astronomical imaging". In: *Astronomy and Astrophysics* 436 (2005), pp. 373–378.

[93] A. Rahimi and B. Recht. "Random features for large-scale kernel machines". In: *Advances in neural information processing systems*. 2007, pp. 1177–1184.

[94] A. Ramdas and R. Tibshirani. "Fast and flexible admm algorithms for trend filtering". In: *Journal of Computational and Graphical Statistics* (2015).

[95] N. Rao, P. Shah, and S. Wright. "Forward-Backward Greedy Algorithms for Atomic Norm Regularization". In: *arXiv:1404.5692* (2014).

[96] H. Rauhut. "Random sampling of sparse trigonometric polynomials". In: *Applied and Computational Harmonic Analysis* 22.1 (2007), pp. 16–42.

[97] B. Recht and C. Ré. "Parallel stochastic gradient algorithms for large-scale matrix completion". In: *Mathematical Programming Computation* 5.2 (2013), pp. 201–226.

[98] Y. Rivenson et al. "Deep Learning Microscopy". In: *arXiv* 1.310 (2017). eprint: 1705.04709.

[99] R. T. Rockafellar. *Convex Analysis*. 1970.

[100] S. Rosset, G. Swirszcz, N. Srebro, and J. Zhu. "ℓ_1 Regularization in Infinite Dimensional Feature Spaces". In: (2007). Ed. by N. H. Bshouty and C. Gentile, pp. 544–558.

[101] M. J. Rust, M. Bates, and X. Zhuang. "Sub-diffraction-limit imaging by stochastic optical reconstruction microscopy (STORM)". In: *Nature methods* 3.10 (2006), p. 793.

[102] D. Sage et al. "Quantitative evaluation of software packages for single-molecule localization microscopy". In: *Nature Methods* 12.8 (2015), pp. 717–724.

[103] S. J. Sahl, S. W. Hell, and S. Jakobs. "Fluorescence nanoscopy in cell biology". In: *Nature Reviews Molecular Cell Biology* 18.11 (2017), p. 685.

[104] G. Schiebinger, E. Robeva, and B. Recht. "Superresolution without separation". In: *Computational Advances in Multi-Sensor Adaptive Processing (CAMSAP), 2015 IEEE 6th International Workshop on*. IEEE. 2015, pp. 45–48.

[105] P. Shah, B. Bhaskar, G. Tang, and B. Recht. "Linear System Identification via Atomic Norm Regularization". In: *arXiv:1204.0590* (2012).

[106] A. Shapiro. "Semi-infinite programming, duality, discretization and optimality conditions". In: *Optimization* 58.2 (2009), pp. 133–161.

[107] Y. Shechtman, L. E. Weiss, A. S. Backer, S. J. Sahl, and W. Moerner. "Precise three-dimensional scan-free multiple-particle tracking over large axial ranges with tetrapod point spread functions". In: *Nano letters* 15.6 (2015), pp. 4194–4199.

[108] J. Skaf and S. Boyd. "Techniques for exploring the suboptimal set". In: *Optimization and Engineering* 11.2 (2010), pp. 319–337.

[109] B. K. Sriperumbudur, K. Fukumizu, and G. R. Lanckriet. "Universality, characteristic kernels and RKHS embedding of measures". In: *Journal of Machine Learning Research* 12.Jul (2011), pp. 2389–2410.

[110] I. Steinwart, D. Hush, and C. Scovel. "An explicit description of the reproducing kernel Hilbert spaces of Gaussian RBF kernels". In: *IEEE Transactions on Information Theory* 52.10 (2006), pp. 4635–4643.

[111] R. Sun, E. Archer, and L. Paninski. "Scalable variational inference for super resolution microscopy". In: *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*. Ed. by A. Singh and J. Zhu. Vol. 54. Proceedings of Machine Learning Research. Fort Lauderdale, FL, USA: PMLR, 2017.

[112] G. Tang, B. N. Bhaskar, P. Shah, and B. Recht. "Compressed sensing off the grid". In: *IEEE Trans. Inf. Thy* 59.11 (2013), pp. 7465–7490.

[113] G. Tang, B. N. Bhaskar, and B. Recht. "Sparse recovery over continuous dictionaries: just discretize". In: *Signals, Systems and Computers, 2013 Asilomar Conference on.* IEEE. 2013, pp. 1043–1047.

[114] R. Tibshirani. "Adaptive piecewise polynomial estimation via trend filtering". In: *The Annals of Statistics* 42.1 (2014), pp. 285–323.

[115] R. Tibshirani. "Regression Shrinkage and Selection Via the Lasso". In: *Journal of the Royal Statistical Society, Series B* 58 (1994), pp. 267–288.

[116] R. J. Tibshirani. "A General Framework for Fast Stagewise Algorithms". In: *arXiv preprint arXiv:1408.5801* (2014).

[117] Z. Ulanowski, Z. Wang, P. H. Kaye, and I. K. Ludlow. "Application of neural networks to the inverse light scattering problem for spheres." In: *Applied optics* 37.18 (1998), pp. 4027–33.

[118] E. Van den Berg and M. P. Friedlander. "Sparse optimization with least-squares constraints". In: *SIAM Journal on Optimization* 21.4 (2011), pp. 1201–1229.

[119] G. Wahba. *Spline Models for Observational Data*. Vol. 59. CBMS-NSF Regional Conference Series in Applied Mathematics. Philadelphia, PA: Society for Industrial and Applied Mathematics (SIAM), 1990.

[120] Q. Wei, K. Fan, L. Carin, and K. A. Heller. "An inner-loop free solution to inverse problems using deep neural networks". In: *arXiv* Nips (2017), pp. 1–20. eprint: `1709.01841`.

[121] M. Weigert et al. “Content-Aware Image Restoration: Pushing the Limits of Fluorescence Microscopy”. In: *bioRxiv* (2017), p. 236463. eprint: 236463 (10.1101).

[122] K. Zhang, W. Zuo, S. Gu, and L. Zhang. “Learning deep CNN denoiser prior for image restoration”. In: *arXiv* (2017).

[123] S. Zhang and E. Salari. “Image Denoising using a Neural Network Based Non-Linear Filter in Wavelet Domain”. In: *Proceedings. (ICASSP ’05). IEEE International Conference on Acoustics, Speech, and Signal Processing, 2005.* Vol. 2. IEEE, 2005, pp. 989–992.

[124] X. Zhang, Y. Yu, and D. Schuurmans. “Accelerated Training for Matrix-norm Regularization: A Boosting Approach”. In: *Advances in Neural Information Processing Systems 26 (NIPS)*. 2012.

[125] L. Zhu, W. Zhang, D. Elnatan, and B. Huang. “Faster STORM using compressed sensing”. In: *Nature methods* 9.7 (2012), pp. 721–723.