Convex Analysis for Minimizing and Learning Submodular Set Functions

Thesis by

Peter Stobbe

In Partial Fulfillment of the Requirements
for the Degree of
Doctor of Philosophy

California Institute of Technology
Pasadena, California

2013
(Defended May 16, 2013)

To Aura

Abstract

The connections between convexity and submodularity are explored, for purposes of minimizing and learning submodular set functions.

First, we develop a novel method for minimizing a particular class of submodular functions, which can be expressed as a sum of concave functions composed with modular functions. The basic algorithm uses an accelerated first order method applied to a smoothed version of its convex extension. The smoothing algorithm is particularly novel as it allows us to treat general concave potentials without needing to construct a piecewise linear approximation as with graph-based techniques.

Second, we derive the general conditions under which it is possible to find a minimizer of a submodular function via a convex problem. This provides a framework for developing submodular minimization algorithms. The framework is then used to develop several algorithms that can be run in a distributed fashion. This is particularly useful for applications where the submodular objective function consists of a sum of many terms, each term dependent on a small part of a large data set.

Lastly, we approach the problem of learning set functions from an unorthodox perspective—sparse reconstruction. We demonstrate an explicit connection between the problem of learning set functions from random evaluations and that of sparse signals. Based on the observation that the Fourier transform for set functions satisfies exactly the conditions needed for sparse reconstruction algorithms to work, we examine some different function classes under which uniform reconstruction is possible.

Contents

List of Figures

List of Algorithms

Chapter 1

Introduction

There is no doubt that convex optimization has proven to be an invaluable tool throughout all of the applied sciences and engineering. Consider though: the formal definition of convexity is a completely abstract concept, yet somehow has proven to be key in the development of numerical algorithms for countless real-world applications. Given the tremendous track record of such a powerful abstract idea, the mandate of the applied mathematics community must be then to attempt to answer the question: "Can convexity be generalized? Can we discover similar abstract concepts that hold the key for solving new, important problems?" And while one search direction of this quest is to look into the realm of continuous functions for quasiconvex or invex functions as generalizations, the other is to look for discrete analogues of convexity. Indeed, many different such discrete generalizations have been discovered [Mur03], yet none of them could accurately be described as a perfect mirror image of convexity. But we would claim that there is one such concept from discrete optimization that in recent years has proven to be the most similar to convexity, not only in terms of its salient abstract features, but also its empirical problem solving utility: *submodularity*.

Submodularity is a property of set functions—meaning functions of some subset of a finite set of objects. It has numerous equivalent definitions, but perhaps the easiest to parse is the following inequality. It says that the change in value of the set function when adding a particular element must be smaller when adding it to a larger[1] set:

$$f(A \cup \{e\}) - f(A) \geq f(B \cup \{e\}) - f(B), \text{ for all elements } e \text{ and sets } A, B \text{ such that } e \notin B \supseteq A.$$

One amazing property of submodularity is that there are powerful algorithms for both maxi-

[1]By "larger" we mean a superset, not just any set of greater cardinality!

mization and minimization with provable guarantees. While both are important, we focus on the latter, as it is in this domain that the connections with convexity are most pronounced. The exact minimum of a submodular function can be found in strongly polynomial time [IFF01]. This is similar to the fact that convex functions are hard to maximize, but easy to minimize (assuming the convex sets which characterize the problem are not too esoteric). This similarity is is not just a coincidence—the connection between submodularity and convexity goes beyond mere analogy. In fact, submodular functions give rise to a convex interpolant (the Lovász extension) that can be used to minimize the submodular function itself. That is, submodular minimization can be cast as a special type of convex minimization. From a dual perspective, every submodular function corresponds to a convex set, called the base polytope, with a very unique and subtle property. The base polytope is defined by exponentially many inequalities, and yet despite that, one can can find the set of maximizing vertices with respect to any linear function without having to compute any inner products. The resulting exposed face of the base polytope depends only on the *order* of the components of the linear function.

Clearly, the connections between submodularity and convexity run deep, and have been known for some time, dating back to at least 1981 with the work of Grötschel, Lovász, and Schrijver [GLS81, Lov83]. Base polytopes (the union of the faces of a polymatroid) were discovered by Edmonds even earlier [Edm70]. Our main goal in this thesis is to explore these connections, but with insight gained from recent advances in convex analysis and sparse reconstruction.

The specific problem that we address, for most of the thesis, is submodular minimization. Despite the known existence of polynomial-time submodular minimization algorithms, the best known exact techniques require a number of function evaluations on the order of n^5 [IO09], where n is the number of variables in the problem. This renders these algorithms impractical for many real-world problems. However, there are various special cases of submodular functions which admit extremely efficient minimization algorithms. One of our objectives in this work has been to bridge the gap between these two extremes of submodular functions. As such, much of our work presented in this thesis has been aimed at developing minimization algorithms for submodular functions which have enough structure to be amenable to optimization, but are not so restrictive that they have little modeling power. That is, we explore the trade-off between specificity and generality of algorithms and submodular function classes, not unlike the trade-off of generalization in statistics and learning. So in this way, the last major part of the thesis is related to what precedes it. Therein we examine a learning problem, that of set functions. We consider submodular functions in particular as a class of objects to learn, but what sets this work apart from classical research on learning set functions is that we use the tools of convex analysis and sparse recovery.

1.1 Main Contributions

In Chapter 4, we develop a novel method for minimizing a particular class of submodular functions, which can be expressed as a sum of concave functions composed with modular functions. The basic algorithm uses an accelerated first order method applied to a smoothed version of the Lovász extension. The smoothing algorithm is particularly novel as it allows us to treat general concave potentials without needing to construct a piecewise linear approximation as with graph-based techniques. This is a fully expanded version of work presented previously [SK10], which did not originally contain the fast method of smoothing.

In Chapter 5, our main technical contribution is elucidating the conditions under which it is possible to find a minimizer of a submodular function via a convex problem. In general one minimizes the Lovász extension together with a separable barrier function, and our Theorem 5.4 gives the weakest conditions yet presented in the literature to guarantee that the convex problem gives a minimizer of the submodular problem; we demonstrate why these conditions are necessary, given some mild assumptions. This provides a general framework for developing submodular minimization algorithms. The framework is then used to develop several algorithms that can be run in a distributed fashion. This is particularly useful for applications where the submodular objective function consists of a sum of many terms, each term dependent on a small part of a large data set.

In Chapter 6 we approach the problem of learning set functions from an unorthodox perspective—sparse reconstruction. We demonstrate an explicit connection between the problem of learning set functions from random evaluations and that of sparse signals. Based on the observation that the Fourier transform for set functions satisfies exactly the conditions needed for sparse reconstruction algorithms to work, we examine some different function classes under which uniform reconstruction is possible. In particular, given the values of the cut function of a graph, we show the graph can be reconstructed. Furthermore, we show

how the assumption of submodularity can be encoded as a constraint of a third order set function and its utility in the reconstruction of a set function.

1.2 Outline of Thesis

- In Chapter 2 we review some concepts from convex analysis and introduce the notation relevant to convex functions and sets.
- Chapter 3 is dedicated to the background material on submodular set functions. Section 3.2 is devoted to the theory of general set functions, whereas in Section 3.3, we review properties of submodular set functions, deriving several of the main relations to convexity. In Section 3.4 we discuss submodular minimization.
- In Chapter 4 we describe our Smoothed Lovász Gradient algorithm for the minimization of decomposable submodular functions. Section 4.4 is dedicated to the classification and representation of decomposable functions. This was originally presented in [SK10].
- In Chapter 5, we develop consensus algorithms for distributed submodular minimization. In Section 5.2, we detail the relationship between convex optimization problems and the minimizers of a submodular function.
- In Chapter 6, we apply the tools of convex analysis and sparse approximation to the problem of learning set functions. This was originally presented in [SK12].

Chapter 2

Background of Convex Analysis

2.1 Basic Concepts

Throughout this thesis, we will use the convention of convex analysis [Roc70, HUL93, AT03] that functions are defined everywhere in $\mathbb{R}^n$, but functions can equal $+\infty$. When we refer to the domain of a function, we mean the set of points where it is finite valued:

$$\mathrm{dom}(f) := \{\, \mathbf{x} \in \mathbb{R}^n \mid f(\mathbf{x}) < +\infty \,\}$$

We define the indicator function of a set to be the function that has that set as its domain, and equals zero there:

$$\delta_C(\mathbf{x}) := \begin{cases} 0 & \mathbf{x} \in C \\ +\infty & \mathbf{x} \notin C \end{cases} \tag{2.1.1}$$

This means that there is no loss of generality in treating constrained optimization problems as unconstrained and vice-versa. When we discuss solving $\min_{\mathbf{x}\in C} f(\mathbf{x})$, where C is some subset of $\mathbb{R}^n$, it is equivalent to solving the unconstrained problem $\min_{\mathbf{x}} f(\mathbf{x}) + \delta_C(\mathbf{x})$. (Clearly we can treat an unconstrained problem as a constrained one with empty constraints.) Convex sets are those which contain every line segment connecting any pair of points in the set. The convex hull of a set is defined as the intersection of all convex sets containing it: $\mathrm{conv}\, S := \bigcap_{\substack{C \subseteq S,\\ C \text{ convex}}} C$.

We can define convex functions in terms of convex sets. The epigraph of a function f on $\mathbb{R}^n$ is the set of points in $\mathbb{R}^{n+1}$ that lie above its graph $\mathrm{epi} f = \{\, (\mathbf{x}, t) \in \mathbb{R}^{n+1} \mid \mathbf{x} \in \mathrm{dom}\, f,\ f(\mathbf{x}) \le t \,\}$. Convex functions are defined as those with a convex epigraph. Convex functions are not necessarily differentiable, but instead have subgradients. For convex f, we define the subdifferential to be the set valued map $\partial f : \mathbb{R}^n \to 2^{\mathbb{R}^n}$ which is the set of subgradients: linear functionals which bound the function from below.

$$\partial f(\mathbf{x}) := \{\boldsymbol{\lambda} \in \mathbb{R}^n \mid f(\mathbf{y}) \geq f(\mathbf{x}) + \langle \mathbf{y} - \mathbf{x}, \boldsymbol{\lambda} \rangle \text{ for all } \mathbf{y} \in \mathbb{R}^n\}$$

Since it is an intersection of half-spaces, the subdifferential is always closed and convex, but it might be empty, even if the function is convex and finite-valued at that point. At any point where a convex function is differentiable, the gradient at that point is the unique subgradient: $\partial f(\mathbf{x}) = \{\nabla f(\mathbf{x})\}$. Subdifferentials are linear and positive homogenous, where addition is in the sense of a Minkowski sum:

$$\partial(f_1 + \alpha f_2) = \partial f_1 + \alpha \, \partial f_2, \; \alpha \geq 0$$

A key tool in analyzing the dual of a convex program is the convex conjugate, also known as the Legendre-Fenchel transform of a function. It is defined as:

$$f^*(\boldsymbol{\lambda}) := \sup_{\mathbf{x} \in \mathbb{R}^n} \langle \mathbf{x}, \boldsymbol{\lambda} \rangle - f(\mathbf{x})$$

This is always convex even if f is not. It is immediate from the definition that:

$$f(\mathbf{x}) + f^*(\boldsymbol{\lambda}) \geq \langle \mathbf{x}, \boldsymbol{\lambda} \rangle \text{ for all } \mathbf{x}, \boldsymbol{\lambda} \in \mathbb{R}^n \tag{2.1.2}$$

Furthermore, note that if $\boldsymbol{\lambda}$ is a subgradient of f at $\mathbf{x}$, then $\langle \mathbf{y}, \boldsymbol{\lambda} \rangle - f(\mathbf{y}) \geq \langle \mathbf{x}, \boldsymbol{\lambda} \rangle - f(\mathbf{x})$ for all $\mathbf{y} \in \mathbb{R}^n$. So Equation 2.1.2 holds with equality if and only if $\boldsymbol{\lambda} \in \partial f(\mathbf{x})$.

We denote the set of convex, proper, lower semi-continuous functions as $\overline{\text{Conv}}$. These functions obey the useful property that they are equal to their biconjugate.

$$f \in \overline{\text{Conv}} \Leftrightarrow f^{**} = f \tag{2.1.3}$$

So, for $f \in \overline{\text{Conv}}$, we can make a stronger statement about when Equation 2.1.2 holds with equality:

$$f(\mathbf{x}) + f^*(\boldsymbol{\lambda}) = \langle \mathbf{x}, \boldsymbol{\lambda} \rangle \quad \Leftrightarrow \quad \mathbf{x} \in \partial f^*(\boldsymbol{\lambda}) \quad \Leftrightarrow \quad \boldsymbol{\lambda} \in \partial f(\mathbf{x})$$

One interpretation of Equation 2.1.3 is that all $f \in \overline{\text{Conv}}$ admit the following sort of self-description:

$$f(\mathbf{x}) = \sup_{\mathbf{y},\boldsymbol{\lambda}} \; f(\mathbf{y}) + \langle \mathbf{x}-\mathbf{y}, \boldsymbol{\lambda}\rangle \quad \text{s.t. } \boldsymbol{\lambda} \in \partial f(\mathbf{y})$$

In Section 3.3, we show that submodular functions enjoy an analogous property.

The fundamental result for expressing the conjugate of a sum of functions is Fenchel's theorem.

Theorem 2.1 (Fenchel Duality). *Suppose the functions $f_i \in \overline{\text{Conv}}$ satisfy $\bigcap_i \operatorname{relint} \operatorname{dom} f_i \neq \emptyset$.*

$$\min_{\mathbf{x}\in\mathbb{R}^n} \sum_i f_i(\mathbf{x}) = \max_{\boldsymbol{\lambda}_i\in\mathbb{R}^n} \; -\sum_i f^*(\boldsymbol{\lambda}_i) \quad s.t. \; \sum_i \boldsymbol{\lambda}_i = \mathbf{0} \tag{2.1.4}$$

Furthermore, the arguments $(\mathbf{x}^*, \boldsymbol{\lambda}_i^*)$ are optimal for the above problems if and only if:

$$\boldsymbol{\lambda}_i^* \in \partial f(\mathbf{x}^*), \qquad \mathbf{x}^* \in \partial f_i^*(\boldsymbol{\lambda}_i^*). \tag{2.1.5}$$

Relation with Lagrangian Duality. We reformulate as a constrained problem, form the Lagrangian of the constrained problem, and then minimize the Lagrangian to obtain a bound valid for all $\boldsymbol{\lambda}_i \in \mathbb{R}^n$:

$$\min_{\mathbf{x}\in\mathbb{R}^n} \sum_i f_i(\mathbf{x}) \geq \min_{\mathbf{x},\mathbf{x}_i\in\mathbb{R}^n} \sum_i f_i(\mathbf{x}_i) + \langle \mathbf{x}-\mathbf{x}_i, \boldsymbol{\lambda}_i\rangle = \begin{cases} -\sum_i f_i^*(\boldsymbol{\lambda}_i) & \text{if } \sum_i \boldsymbol{\lambda}_i = \mathbf{0} \\ -\infty & \text{if } \sum_i \boldsymbol{\lambda}_i \neq \mathbf{0} \end{cases}$$

Note that by assumption, the dual problem is feasible, so the bound involves finite numbers. Likewise, for all $\mathbf{x} \in \mathbb{R}^n$, we have:

$$\max_{\sum \boldsymbol{\lambda}_i = \mathbf{0}} -\sum_i f^*(\boldsymbol{\lambda}_i) \leq \max_{\boldsymbol{\lambda}_i\in\mathbb{R}^n} \sum_i \langle \mathbf{x}, \boldsymbol{\lambda}_i\rangle - \sum_i f^*(\boldsymbol{\lambda}_i) = \sum_i f_i^{**}(\mathbf{x})$$

Since $f_i^{**} = f_i$, this means that the theorem gives conditions under which strong duality holds, and a mechanical formula for deriving the dual program.

Another way to state Fenchel's theorem is through infimal convolutions. We denote the infimal convolution of functions with the symbol $\square$ and define it by the following formula:

$$f \square g(\mathbf{x}) := \inf_{\mathbf{y}\in\mathbb{R}^n} f(\mathbf{x}-\mathbf{y}) + g(\mathbf{y})$$

If f and g satisfy conditions sufficient for Fenchel's duality theorem to hold, then infimal convolution is essentially the operation which is dual to addition under the Legendre-Fenchel transform: $(f \square g)^* = f^* + g^*$ and $(f+g)^* = f^* \square g^*$.

2.2 Proximal Operators

A key step in many convex minimization algorithms is solving for the infimal convolution of an objective function with a quadratic function at the current iterate. For any function $f \in \overline{\text{Conv}}$, we define its proximal operator or 'prox' as:

$$\text{prox}_f(\mathbf{x}) := \arg\min_{\mathbf{y}\in\mathbb{R}^n} f(\mathbf{y}) + \frac{1}{2}\|\mathbf{x}-\mathbf{y}\|^2 \tag{2.2.1}$$

For a thorough explanation of the proximal operator, see [CP11]. We review a few important ideas and identities.

By strong convexity of the quadratic term, the minimum is a unique point so the prox is well-defined. (In its most general form, one can use any Bregman distance [Brè67] in place of the quadratic, but the present definition is sufficient for our purposes.) The point $\mathbf{p} = \text{prox}_f(\mathbf{x})$ is uniquely determined by the optimality conditions given by subgradients:

$$\mathbf{p} \in \mathbf{x} - \partial f(\mathbf{p}) \tag{2.2.2}$$

One way to interpret this equation is that the proximal operator performs an implicit gradient descent step. That is, a basic gradient descent method for convex minimization might use an explicit update rule such as: $\mathbf{x}^{k+1} = \mathbf{x}^k - \epsilon\nabla f(\mathbf{x}^k)$, where f is a convex differentiable objective function, $\mathbf{x}^k$ is the iterate at step k, and ϵ is some step size. Suppose instead we use an *implicit* update rule: $\mathbf{x}^{k+1} = \mathbf{x}^k - \epsilon\nabla f(\mathbf{x}^{k+1})$. This update rule is actually a proximal operator—by Equation 2.2.2, it is equivalent to $\mathbf{x}^{k+1} = \text{prox}_{\epsilon f}(\mathbf{x}^k)$. So in this sense, the prox is a form of implicit numerical integration of the dynamical system $\dot{\mathbf{x}} = -\nabla f$, but it can be applied to nondifferentiable convex functions.

From another point of view, the proximal operator is a generalization of projection onto a convex set. When the function f in Equation 2.2.1 is the indicator function of a closed convex set (as defined in Equation 2.1.1), then the prox is exactly the projection operator, which we denote with the letter Π.

$$\Pi_C(\mathbf{x}) := \operatorname*{arg\,min}_{\mathbf{y}\in C} \|\mathbf{x}-\mathbf{y}\|$$

That is, $\mathrm{prox}_{\delta_C}(\mathbf{x}) = \Pi_C(\mathbf{x})$.

If we apply Fenchel's theorem to the proximal problem, we get an important identity relating the prox of a function with the prox of its conjugate. By Equation 2.1.4, we have for all $f \in \overline{\mathrm{Conv}}$:

$$\min_{\mathbf{y}\in\mathbb{R}^n} \frac{1}{2}\|\mathbf{x}-\mathbf{y}\|^2 + f(\mathbf{y}) = \max_{\boldsymbol{\lambda}\in\mathbb{R}^n} -\frac{1}{2}\|\boldsymbol{\lambda}\mathbf{x}\|^2 - \langle \mathbf{x}, \boldsymbol{\lambda}\rangle - f^*(-\boldsymbol{\lambda})$$

Furthermore, by Equation 2.1.5, the optimal arguments satisfy $\boldsymbol{\lambda}^* = \mathbf{y}^* - \mathbf{x}$, $\mathbf{y}^* \in \partial f^*(-\boldsymbol{\lambda}^*)$, $-\boldsymbol{\lambda}^* \in \partial f(\mathbf{y}^*)$, so by Equation 2.2.2, we have $\mathbf{y}^* = \mathrm{prox}_f(\mathbf{x})$ and $-\boldsymbol{\lambda}^* = \mathrm{prox}_{f^*}(\mathbf{x})$. We conclude that:

$$\mathrm{prox}_f(\mathbf{x}) + \mathrm{prox}_{f^*}(\mathbf{x}) = \mathbf{x} \tag{2.2.3}$$

The practical implication of this, from a computational standpoint, is that computing the prox of a function is of the same complexity as computing the prox of its conjugate function.

In particular, consider convex indicator functions and their conjugates, which we denote with the letter σ. These are called the support functions of a set:

$$\sigma_C(\mathbf{x}) := \sup_{\boldsymbol{\lambda}\in C} \langle \mathbf{x}, \boldsymbol{\lambda}\rangle$$

We assume C is a closed convex set, so $\delta_C^{**} = \sigma_C^* = \delta_C$. By Equation 2.2.3, we can compute the prox for σ_C by projecting onto C.

$$\mathrm{prox}_{\sigma_C}(\mathbf{x}) = \mathbf{x} - \Pi_C(\mathbf{x})$$

It is clear by definition that support functions are positive homogenous: $\beta\sigma_C(\mathbf{x}) = \sigma_C(\beta\mathbf{x})$ for $\beta > 0$. This means that the proximal operators of σ_C obey a scaling property that other convex functions do not: $\mathrm{prox}_{\beta\sigma_C}(\mathbf{x}) = \beta\,\mathrm{prox}_{\sigma_C}(\mathbf{x}/\beta) = \mathbf{x} - \beta\Pi_C(\mathbf{x}/\beta)$.

An important special case of a support function is when C is symmetric about the origin and has nonempty interior; if this is true, then σ_C is a norm and C is the unit ball of the corresponding dual norm. For example, in the context of sparse approximation, one often minimizes the ℓ_1 norm to promote sparsity of a vector. The proximal step thus involves subtracting off a projection onto the ℓ_∞ ball, which is equivalent to soft thresholding the components of a vector.

Another interesting case is when the support function of a convex set is itself an indicator function of another convex set. It is not hard to see that this is true if and only if the sets are cones. A set K is a cone if it contains all positive multiples of itself: $\beta K \subseteq K$ for any $\beta > 0$. The corresponding polar cone K° is defined as the set of the dual vectors which have nonpositive inner product for all points in the cone: $K^\circ := \{\boldsymbol{\lambda} \in \mathbb{R}^n \mid \langle \mathbf{x}, \boldsymbol{\lambda} \rangle \leq 0 \text{ for all } \mathbf{x} \in K\}$. If K is also closed and convex, then $\delta_K \in \overline{\text{Conv}}$, and we have $\delta_K^* = \sigma_K = \delta_{K^\circ}$. So by Equation 2.2.3, we can rederive a basic result in conic analysis—for any closed convex cone, every point in $\mathbb{R}^n$ is uniquely decomposed as a sum of a point in the cone and a point in the polar cone: $\mathbf{x} = \Pi_K(\mathbf{x}) + \Pi_{K^\circ}(\mathbf{x})$.

Prox of a Sum. In general, there is no way to combine and simplify expressions for proximal operators in closed form. However, we can get an implicit characterization of the proximal operator of a sum of functions.

Proposition 2.2. *Suppose g is a positive combination of convex functions. Specifically, let $g := \sum_i \omega_i f_i$ where $\sum_i \omega_i = \Omega$, $\omega_i > 0$, and $f_i \in \overline{\text{Conv}}$. Then $\mathbf{x} = \text{prox}_g(\mathbf{y})$ if and only if there are dual vectors $\boldsymbol{\lambda}_i$ such that:*

$$\sum_i \omega_i \boldsymbol{\lambda}_i = \mathbf{0} \tag{2.2.4}$$

$$\mathbf{x} = \text{prox}_{\Omega f_i}(\mathbf{y} + \boldsymbol{\lambda}_i) \quad \textit{for all } i \tag{2.2.5}$$

Proof. To show the forward direction, note that if $\mathbf{x} = \text{prox}_g(\mathbf{y})$, optimality implies:

$$\mathbf{0} \in \mathbf{x} - \mathbf{y} + \partial g(\mathbf{x}) = \frac{1}{\Omega} \sum_i \omega_i \left(\mathbf{x} - \mathbf{y} + \Omega\, \partial f_i(\mathbf{x}) \right)$$

By linearity of subgradients, there must exist $\boldsymbol{\lambda}_i$ satisfying Equation 2.2.4 such that

$$\boldsymbol{\lambda}_i \in \mathbf{x} - \mathbf{y} + \Omega\, \partial f_i(\mathbf{x}) \quad \text{for all } i$$

By Equation 2.2.2, this is exactly the optimality condition needed to imply $\mathbf{x} = \text{prox}_{\Omega f_i}(\mathbf{y}+\boldsymbol{\lambda}_i)$.

To show the backward direction, note that if $\mathbf{x}$ and $\boldsymbol{\lambda}_i$ satisfy Equation 2.2.4 and Equation 2.2.5, then

$$\begin{aligned} \mathbf{x} &= \operatorname*{arg\,min}_{\mathbf{z}\in\mathbb{R}^n} \frac{1}{2\Omega}\|\mathbf{z}-(\mathbf{y}+\boldsymbol{\lambda}_i)\|^2 + f_i(\mathbf{z}) \quad \text{for all } i \\ &= \operatorname*{arg\,min}_{\mathbf{z}\in\mathbb{R}^n} \sum_i \omega_i \left[\frac{1}{2\Omega}\|\mathbf{z}-(\mathbf{y}+\boldsymbol{\lambda}_i)\|^2 + f_i(\mathbf{z})\right] \\ &= \operatorname*{arg\,min}_{\mathbf{z}\in\mathbb{R}^n} \frac{1}{2}\|\mathbf{z}-\mathbf{y}\|^2 + g(\mathbf{z}) = \text{prox}_g(\mathbf{y}) \end{aligned} \tag{2.2.6}$$

Hence, $\mathbf{x}$ minimizes each individual term of the sum on the right hand side of Equation 2.2.6, so therefore $\mathbf{x}$ must minimize the sum of those terms. Within the sum, the dual variables $\boldsymbol{\lambda}_i$ cancel each other out. $\square$

Chapter 3

Set Functions and Submodularity

3.1 Overview

In Section 3.2, we review some basic concepts from the study of general set functions (not necessarily submodular). We also introduce notation related to set functions that we use throughout the thesis. In subsection 3.2.1, we define the set function derivative in a way that emphasizes the symmetric shift operator. In subsection 3.2.2, we define and discuss monotonic functions of general order; this is the natural generalization of submodular to higher order differences. In subsection 3.2.3, we define the Fourier transform for set functions, a key tool in learning theory of set functions. In subsection 3.2.4, we define other linear tranforms of variables and show the common connection between them in terms of tensor products.

Finally in Section 3.3, we focus on submodularity in detail. As this material may not be nearly as well-known outside of specialists in combinatorial optimization, we attempt to be more thorough by proving some of the key relationships between submodularity and convexity.

In Section 3.4, we specifically review the problem of submodular minimization, which is the primary subject of the Chapter 4 and Chapter 5. We derive equivalent of a duality gap, and then review some of the existing algorithms.

3.2 General Set Functions

While convex analysis provides much of our notation and terminology, our main object of study in this work are functions over the Boolean cube $\mathbb{Z}_2^n = \{0,1\}^n$. The term Boolean function refers to functions on the cube which take on Boolean values; real-valued functions of Boolean vectors are a generalization called pseudoboolean functions. However, since the power set of a finite set is equivalent to the Boolean cube, in other technical literature the term set function is used. We will use the terminology and notation of sets and set functions, which is more common when the functions are submodular. However, it will be useful to use double brackets $[\![\ ]\!]$ to denote the Boolean value of some statement:

$$[\![\text{statement}]\!] := \begin{cases} 1 & \text{if the statement is true} \\ 0 & \text{if the statement is false} \end{cases}$$

Throughout this thesis, we work in some context where there is some finite ground set E of cardinality n. We let $\mathcal{H} = \mathbb{R}^{2^E}$ be the space of real-valued functions on subsets of E. That is, if $f \in \mathcal{H}$ then it is a function $f : 2^E \to \mathbb{R}$, where 2^E is the power set of E. We use the upper-case letters for subsets of E, and lowercase letters for elements of E. Also, we drop brackets for small sets: denoting b or bc rather than $\{b\}$ or $\{b, c\}$ when the context is clear. We occasionally use $+$ to mean union, as in $A + b + c = A \cup \{b\} \cup \{c\}$, but only when the sets are disjoint. When we say that a collection of sets is disjoint, we mean they are pairwise disjoint.

We treat the elements of E as the indices of vectors in $\mathbb{R}^n$. Then for any $A \in 2^E$, we define $\mathbb{1}_A \in \mathbb{R}^n$ to be the indicator vector of that set.

$$\mathbb{1}_A[e] := [\![e \in A]\!] = \begin{cases} 1 & \text{if } e \in A \\ 0 & \text{if } e \in E \setminus A \end{cases}$$

For example, $\{\mathbb{1}_e\}_{e \in E}$ is the set of standard unit vectors. Clearly 2^E is isomorphic to the commutative group $\mathbb{Z}_2^n$ under the mapping of indicator vectors: $\mathbb{1}_A + \mathbb{1}_B \equiv \mathbb{1}_{A \ominus B} \mod 2$. That is, addition over the group 2^E is the symmetric set difference ($\ominus$), defined by:

$$A \ominus B := (A \setminus B) \cup (B \setminus A)$$

There are some straightforward consequences of this isomorphism. For example, since $\emptyset$ acts as the identity element, and every element of the group is of order two, $A \ominus B = \emptyset$ if and only if $A = B$. Furthermore, the group of characters of $\mathbb{Z}_2^n$ is used to define the Fourier transform for set functions, as shown in subsection 3.2.3.

3.2.1 Set Function Derivatives

We now define linear operators on set functions in a way that parallels conventions in, for example, signal processing; this is a slightly different approach than that usually seen in the literature on set functions. We define the (symmetric) shift operator $S_B : \mathcal{H} \to \mathcal{H}$ as the operator which applies a symmetric difference to the argument of a set function:

$$S_B f(A) := f(A \ominus B)$$

If we shift with respect to a singleton we write $S_b = S_{\{b\}}$. Clearly these operators are linear and are isomorphic to the group $\mathbb{Z}_2^n$, meaning that $S_B S_C = S_{B \ominus C}$. Hence, the operators commute since $\mathbb{Z}_2^n$ is commutative. We define the discrete derivative with respect to a single element as the difference between a shift operator of the element and the identity.

$$\Delta_b := S_b - 1$$

Note that the derivative operator has the same function space $\mathcal{H}$ as its domain and range. A common definition of discrete derivative uses union and intersection rather than symmetric difference, but the advantage of using the symmetric difference is that it is diagonalized by the Fourier transform (i.e., it is a convolution), but the operator defined in terms of unions is not. In any case, the interpretation of the derivative is straightforward; if we evaluate $\Delta_b f$ on sets that do not contain the element b, we get the change in value of the function due to adding the element:

$$\Delta_b f(A) = f(A \ominus b) - f(A)$$

So if f is interpreted as a valuation function, then $\Delta_b f(A)$ is the marginal value of item b relative to the set A. If we evaluate on a set that already contains b, then the derivative is the change in value from removing the set, which is -1 times the marginal value.

We define the derivative operator with respect to a set B as the product of derivative operators with respect to each element in the set:

$$\Delta_B := \prod_{b\in B} \Delta_b = \prod_{b\in B}(S_b - 1) \tag{3.2.1}$$

Because the shift operators commute, this equation does not depend on the ordering of the product, so the above expression for derivative is well-defined. As is the standard convention for empty products, we define $\Delta_\emptyset$ to be the identity. It is obvious from our definition that derivatives with respect to disjoint sets combine to form the derivative with respect to their union. That is, assuming B, C disjoint,[1] we have:

$$\Delta_B \Delta_C = \prod_{b\in B} \Delta_b \prod_{c\in C} \Delta_c = \prod_{e\in B\cup C} \Delta_e = \Delta_{B\cup C}$$

If we expand out the terms in the product in Equation 3.2.1, we can get an equivalent definition of the derivative as a sum of shift operators:

$$\Delta_B = \sum_{C\in 2^E} [\![C \subseteq B]\!]\,(-1)^{|B|+|C|} S_C$$

For example, the derivative with respect to a pair of elements $\Delta_{bc} = \Delta_{\{b,c\}}$ equals:

$$\Delta_{bc} f(A) = f(A \ominus bc) - f(A \ominus b) - f(A \ominus c) + f(A)$$

This also called the second order difference operator. Note that there is also a product rule for discrete derivatives:

Lemma 3.1 (Product Rule for Set Function Derivatives)**.**

$$\Delta_B(fg) = \sum_{C\in 2^E} [\![C \subseteq B]\!]\,(\Delta_C f)(S_C \Delta_{B\setminus C}\, g) \tag{3.2.2}$$

Proof. When $|B| = 1$, this is true by the following identity:

$$\begin{aligned}\Delta_b(fg) &= (S_b f - f) S_b g + f(S_b g - g)\\ &= (\Delta_b f)(S_b \Delta_\emptyset g) + (\Delta_\emptyset f)(S_\emptyset \Delta_b g)\end{aligned}$$

[1] Since $(\Delta_b)^2 = -2\Delta_b$, the general formula is $\Delta_B \Delta_C = (-2)^{|B\cap C|} \Delta_{B\cup C}$.

If $|B| > 1$, iterating over all elements $b \in B$ results in Equation 3.2.2. $\square$

3.2.2 Monotone and Low Order Functions

With our definition of derivative, we can define the cones of order q monotone functions, which we denote $\mathcal{H}_q^+$ (resp. $\mathcal{H}_q^-$). It is the subset of functions with all order q derivatives nonnegative (resp. nonpositive). With a slight overload of notation, we will use the symbol $\pm$ rather than state all equations for $\mathcal{H}_q^+$ and $\mathcal{H}_q^-$ separately.

$$\mathcal{H}_q^{\pm} := \{ f \in \mathcal{H} \mid \pm\Delta_B f(A) \geq 0 \text{ for all } A, B \in 2^E \text{ with } A \cap B = \emptyset,\ |B| = q \}$$

The first few of these cones have more common names, and simple alternate characterizations:

- $\mathcal{H}_0^{\pm}$ nonnegative/nonpositive: $\pm f(A) \geq 0$
- $\mathcal{H}_1^{\pm}$ nondecreasing/nonincreasing: $\pm(f(A \cup B) - f(A)) \geq 0$
- $\mathcal{H}_2^{\pm}$ supermodular/submodular: $\pm(f(A \cup B \cup C) - f(A \cup B) - f(A \cup C) + f(A)) \geq 0$

Note that the characterizations we list involve adding and removing entire sets rather than just single elements. This fact generalizes nicely to higher order monotone functions. Informally, the following proposition states that for a q monotone function, we can replace the singleton sets that occur in the definition of the discrete derivative (Equation 3.2.1) with general disjoint sets, and still get a valid inequality.

Proposition 3.2. *The function f is in $\mathcal{H}_q^{\pm}$ if and only if for all collections of $q+1$ disjoints sets $A, B_1, \ldots B_q$, we have:*

$$\pm \prod_{i=1}^{q} \left(S_{B_i} - 1 \right) f(A) \geq 0 \tag{3.2.3}$$

Proof. Clearly if C is an arbitrary subset of size q, disjoint from A, we can choose each of the sets B_i to be singletons such that $C = B_1 + \ldots + B_q$, which implies the operator product in Equation 3.2.3 is the set derivative with respect to C. Hence $f \in \mathcal{H}_q^{\pm}$.

Conversely, if $f \in \mathcal{H}_q^{\pm}$, we wish to prove that Equation 3.2.3 holds for arbitrary choices of disjoint sets. Fix the disjoint sets, and for each set B_i, choose an ordering of its elements, resulting in a chain of strict subsets. That is, let $B(i,0) = \emptyset$ and $B(i, j-1) + c(i, j) := B(i, j) \subseteq$

$B(i)$ for $j = 1 \dots |B_i|$. Then each term in the product can be expressed as a telescoping sum:

$$S_{B_i} - 1 = \sum_{j=1}^{|B_i|} S_{B(i,j)} - S_{B(i,j-1)} = \sum_{j=1}^{|B_i|} S_{B(i,j-1)} \Delta_{c(i,j)}$$

By substituting this equivalence into the each term in the the product from Equation 3.2.3, and then expanding the sum out to $|B_1| \dots |B_q|$ terms, we get:

$$\pm \prod_{i=1}^{q} \left(S_{B_i} - 1\right) f(A) = \pm \prod_{i=1}^{q} \sum_{j=1}^{|B_i|} S_{B(i,j-1)} \Delta_{c(i,j)} f(A) = \sum_{j_1,\dots,j_q} \pm S_{B'(j_1,\dots j_q)} \Delta_{C'(j_1,\dots j_q)} f(A),$$
$$B'(j_1, \dots j_q) := \cup_{i=1}^{q} B(i, j_i - 1), \qquad C'(j_1, \dots j_q) := \{c(i, j_i)\}_{i=1}^{q}.$$

Note that for each term in the series, the argument A, shifting sets B' and derivative sets C' are disjoint. Also each set C' is of size q. Therefore, the expression from Equation 3.2.3 is equivalent to a sum of order q derivatives. Since $f \in \mathcal{H}_q^{\pm}$ implies that order q derivatives are uniformly nonnegative (resp. nonpositive), the entire sum is nonnegative. $\square$

This result is included in [FH05] but given as a symmetric statement over collections of k general sets, not necessarily disjoint. Given general sets $C_1, \dots, C_q$, this defines $q+1$ disjoint sets as follows: $A = \bigcap_{j=1}^{q} C_j$, $B_i = \bigcap_{j \neq i} C_j \setminus C_i$. Then by applying the above Proposition to the disjoint sets, we get the result in the form stated in [FH05]:

$$\pm(-1)^q \left(f\left(\cap_{j=1}^{q} C_j\right) + \sum_{\substack{J \subseteq \{1,\dots,q\} \\ J \neq \emptyset}} (-1)^{|J|} f\left(\cup_{i \in J} \cap_{j \neq i} C_j\right) \right) \geq 0.$$

Products of Monotonic Functions. Due to the product rule of Equation 3.2.2, we can get simple rules for classifying products of functions if they obey certain patterns of monotonicities. To express the following lemma, we will need variables to be signs: $\{+,-\}$. For these purposes they are equivalent to the unit numbers $\{+1,-1\}$.

Lemma 3.3. *Suppose $f, g \in \mathcal{H}$ are monotonic for every order up to order q:*

$$f \in \mathcal{H}_0^{s_0} \cap \dots \cap \mathcal{H}_q^{s_q}, \qquad g \in \mathcal{H}_0^{t_0} \cap \dots \cap \mathcal{H}_q^{t_q}.$$

If the signs of the monotonicities $s_k, t_k \in \{+,-\}$ satisfy $\gamma = s_0 t_q = s_1 t_{q-1} = \dots = s_q t_0$, then

$fg \in \mathcal{H}_q^{\gamma}$.

Proof. Let B be any set of size q. We use the product rule (Equation 3.2.2) to take the derivative of fg with respect to B. For simplicity we do not write the argument A, but assume it is disjoint from B. Within the resulting sum, we use the factorization $\gamma = s_k t_{q-k}$:

$$\begin{aligned}\gamma\Delta_B(fg) &= \sum_{C\in 2^E} [\![C \subseteq B]\!]\,\gamma(\Delta_C f)(S_C \Delta_{B\setminus C} g)\\ &= \sum_{k=0}^{q}\sum_{C\in 2^E} [\![C \subseteq B, |C| = k]\!]\,(s_k \Delta_C f)(t_{q-k} S_C \Delta_{B\setminus C} g)\end{aligned}$$

For $|C| = k$, we have $s_k \Delta_C f \geq 0$ since $f \in \mathcal{H}_0^{s_k}$ and $t_{q-k} S_C \Delta_{B\setminus C} g \geq 0$ since $f \in \mathcal{H}_{q-k}^{t_{q-k}}$ and $|B \setminus C| = q-k$. So each term in the sum multiplies to a nonnegative sign. Thus $\gamma\Delta_B(fg) \geq 0$ for arbitrary B of size q, so indeed $fg \in \mathcal{H}_q^{\gamma}$. □

For example, if f is nonnegative, nondecreasing, submodular ($\mathcal{H}_0^+ \cap \mathcal{H}_1^+ \cap \mathcal{H}_2^-$), and g is nonnegative nonincreasing submodular ($\mathcal{H}_0^+ \cap \mathcal{H}_1^- \cap \mathcal{H}_2^-$), then the product fg is nonnegative submodular $\mathcal{H}_0^+ \cap \mathcal{H}_2^-$, but neither increasing nor decreasing in general. Another example of Lemma 3.3 is that the cone of nonnegative, nondecreasing, supermodular functions ($\mathcal{H}_0^+ \cap \mathcal{H}_1^+ \cap \mathcal{H}_2^+$) is closed under multiplication.

We define q-th order functions, as functions with all derivatives beyond order q equal to zero.

$$\mathcal{H}_q := \{\, f \in \mathcal{H} \mid \Delta_B f(A) = 0 \ \text{ for all } A, B \in 2^E \text{ with } \ |B| > q \,\}$$

It is easy to see from the definition that: $\mathcal{H}_q = \mathcal{H}_{q+1}^+ \cap \mathcal{H}_{q+1}^-$ and $\mathcal{H}_q \subset \mathcal{H}_{q+1}$. The set of zeroth order functions are constant functions. In the context of set functions, first order functions are known as modular, and admit the description: $f(A) = f(\emptyset) + \sum_{a\in A}(f(a) - f(\emptyset))$.

3.2.3 Fourier Analysis of Set Functions

In this section, we briefly introduce the Fourier transform for set functions, but it is primarily expanded upon in Chapter 6. The characters of the group 2^E can be written as $\psi_B(A) := (-1)^{|A\cap B|}$ since $\psi_B(A_1)\psi_B(A_2) = (-1)^{|A_1\cap B|}(-1)^{|A_2\cap B|} = (-1)^{|(A_1\ominus A_2)\cap B|} = \psi_B(A_1 \ominus A_2)$. These oscillate between 1 and -1 when elements of particular set B are added or removed from the argument. Thus, we define the Fourier tranform of a set function by taking the appropriately normalized inner product with such functions:

$$\widehat{f}(B) := \frac{1}{2^{|E|}} \sum_{A \in 2^E} f(A)\psi_B(A)$$

This is an orthogonal transform, so though the inverse formula has a different normalization constant, it is otherwise the same: $f(A) = \sum_{B \in 2^E} \widehat{f}(B)\psi_B(A)$. One way to interpret the Fourier coefficient for a set is that it is the average of all the derivatives with respect to that set (modulo a factor of -1 for odd sets).

$$\widehat{f}(B) = \frac{(-1)^{|B|}}{2^{|E \setminus B|}} \sum_{A \in 2^E} [\![A \subseteq E \setminus B]\!]\, \Delta_B f(A) \tag{3.2.4}$$

Next, we define convolutions of set functions:

$$f * g(A) := \sum_{B \in 2^E} f(A \ominus B) g(B)$$

This satisfies the standard properties of a convolution: Commutivity $f * g = g * f$, Associativity, $(f * g) * h = f * (g * h)$, Linearity $(\alpha f + \beta g) * h = \alpha(f * h) + \beta(g * h)$. Also, convolution in the time domain is multiplication in the Fourier Domain, and vice-versa.

$$\widehat{f * g}(B) = 2^n \widehat{f}(B)\widehat{g}(B), \qquad \widehat{fg}(B) = (\widehat{f} * \widehat{g})(B).$$

The advantage of our definition for set function derivatives is that it is just a convolution. That is, $\Delta_C f = f * g$ where $g(A) = [\![A \subseteq C]\!]\,(-1)^{|A|+|C|}$, and $2^n \widehat{g}(B) = [\![C \subseteq B]\!]\,(-2)^{|C|}$. So a derivative can be treated as a sort of high-pass filter. After taking the derivative with respect to C, all coefficients which are not subsets of C are zeroed out: $\widehat{\Delta_C f}(B) = [\![C \subseteq B]\!](-2)^{|C|}\widehat{f}(B)$. This gives us a formula to express a derivative as a sum of Fourier coefficents:

$$\Delta_C f(A) = (-2)^{|C|} \sum_{B \in 2^E} [\![C \subseteq B]\!]\, \widehat{f}(B)\psi_B(A) \tag{3.2.5}$$

Another important linear operator that can be expressed as a convolution is the projection of a function onto the space of low order functions. By Equation 3.2.4, the subspace $\mathcal{H}_q$ can be characterized as those functions with Fourier transform supported only on sets of size up to

q. That is:

$$\mathcal{H}_q = \{ f \in \mathcal{H} \mid \widehat{f}(B) = 0 \text{ for all } |B| > q \}$$

The best q-th order approximation for a set function in an ℓ_2 sense is given by setting all higher order Fourier coefficients to zero: $g = \arg\min_{g' \in \mathcal{H}_q} \|f - g'\| \iff \widehat{g}(B) = [\![|B| \leq q]\!] \widehat{f}(B)$. This is immediate from the fact that the Fourier transform is an isometry. See [HH92] and [GMR00] for the formulas of this operation in terms of different function bases.

In elementary signal processing, the domains analyzed are the continuous/discrete circle/line ($\mathbb{Z}_n$, $[0,1]$, $\mathbb{Z}$, or $\mathbb{R}$). Exactly as there, (periodic) shifting, derivatives, and low pass filtering can all be expressed as convolutions, so unsurprisingly these operations all commute with each other.

3.2.4 Tensor Product Bases of Set Functions

In addition to the standard basis and the Fourier transform, there are several other useful bases for representing set functions. For a thorough discussion, see [GMR00], but let us review two of the more important ones. The feature common to all the bases that we present is that they can be expressed easily through tensor products.

Möbius Transform and Boolean Polynomials. One way to represent set functions is as a multilinear (a.k.a. polynomial) function over the Boolean cube $p(\mathbf{x}) = p_0 + p_1 x[1] + p_2 x[2] + \ldots p_{1\ldots n} x[1] \ldots x[n]$. This gives a simple way to interpolate a set function continuously over the unit cube by letting the variable $\mathbf{x}$ take on values $[0,1]^n \subset \mathbb{R}^n$. In this case, the interpolation is multilinear by convention since Booleans satisfy $x^2 = x$, and so there is no reason to include nonlinear terms such as $x[i]^2$.

To calculate the coeffcient of a polynomial that represents a set function, we evaluate the set function derivative at the empty set. A set function derivative is exactly the stencil of a mixed derivative of an n dimensional function; an order q discrete derivative is the forward difference operator tensored in some set of q dimensions. Since the Boolean polynomial is multilinear, in general, set function derivatives equal the continuous partial derivatives exactly (provided the argument set and derivative set are disjoint). For example: suppose $E = \{1,2,3\}$, and the set function f is related to the Boolean polynomial p by $f(A) = p(\mathbb{1}_A)$. Then the set function derivative with respect to the set $\{1,2\}$ evaluated at the set $\{3\}$ equals the mixed partial derivative of p at the corresponding corner: $\Delta_{12}f(3) = D_{12}p(\mathbb{1}_3) = p_{12} + p_{123}$.

The transformation from set function values to polynomial coefficients is sometimes called the Möbius transform. We can express this not only as a set derivative, but also we can use Equation 3.2.5 to equate it with a sum of Fourier coefficients.

$$g(B) = \Delta_B f(\emptyset) = (-2)^{|B|} \sum_C [\![B \subseteq C]\!]\, \widehat{f}(C)$$

$$f(A) = \sum_B [\![B \subseteq A]\!]\, g(B)$$

Disjuction. Functions of the form $\min(1, |A \cap B|)$ are equivalent to a disjunction (Boolean OR) operation. A sum of such functions is equivalent to a set cover function, as described in subsection 4.4.2. These functions do not quite form a basis of $\mathcal{H}$, since they all have $f(\emptyset) = 0$. But by including a constant offset, we get a basis for $\mathcal{H}$. The corresponding analysis and synthesis formulas are:

$$h(B) = -\Delta_B f(E) = -2^{|B|} \Delta_{E \setminus B} \widehat{f}(E)$$

$$f(A) = f(\emptyset) + \sum_B \min(1, |A \cap B|) h(B)$$

There is an interesting consequence to the formula relating the set cover coefficients to Fourier coefficients. The original function f is set cover representable (meaning it can be expressed as a nonnegative combination of such functions, so that $h(B) \geq 0$ for all nonempty B) if and only if the Fourier coefficients $\widehat{f}(B)$ for nonempty sets are nonpositive and nondecreasing.

Tensor Product Notation. The above formulas may seem a bit mysterious and hard to check for correctness. However, the calculations are fairly simple if you consider set functions as vectors in $\mathbb{R}^{2^n}$, since all of the bases we have described can be expressed in terms of n-fold tensor products of 2×2 matrices. Let us denote $(\otimes n)$ for the operation of tensoring a matrix with itself n times. Then the main formulas of this section can be written as:

$$\text{Fourier:} \quad \widehat{f} = \begin{bmatrix} 1/2 & 1/2 \\ 1/2 & -1/2 \end{bmatrix}^{\otimes n} f, \qquad f = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}^{\otimes n} \widehat{f}.$$

$$\text{Möbius:} \quad g = \begin{bmatrix} 1 & 0 \\ -1 & 1 \end{bmatrix}^{\otimes n} f, \qquad f = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}^{\otimes n} g.$$

$$\text{Disjunction:} \qquad h = -\begin{bmatrix} 0 & 1 \\ 1 & -1 \end{bmatrix}^{\otimes n} f, \qquad f = f(\emptyset) + \left(\begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}^{\otimes n} - \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}^{\otimes n} \right) h.$$

Therefore, it is straightforward to derive the basic formulas relating different tensor product bases by simply inverting and/or multiplying 2×2 matrices.

3.3 Properties of Submodular Set Functions

There are many equivalent characterizations of submodularity. As we have already introduced the notion of set derivative, we give a few which are directly related to set derivatives:

1. $f(A+b+c) - f(A+b) - f(A+c) + f(A) \leq 0,$ for all $b, c \in E,\ A \subseteq E - b - c$.

2. $f(A \cup B \cup C) - f(A \cup B) - f(A \cup C) + f(A) \leq 0,$ for all A, B, C disjoint.

3. $f(A \cap B) + f(A \cup B) \leq f(A) + f(B)$ for all $A, B \in 2^E$.

4. $f(A+c) - f(A) \geq f(B+c) - f(B)$ for all $c \in E,\ A \subseteq B \subseteq E - c$.

There are $\binom{n}{2} 2^{n-2}$ different inequalities to check in item 1. This is the minimum number of inequalities needed to check submodularity in general. That is, no subset of these inequalities implies any other subset of them. Theorem 3.2 gives the characterizations of items 2. The interpretation of item 4 is that first order differences are decreasing functions. It is an easy consequence of item 3. To test submodularity, it is generally easiest to use item 1. Besides the fact that it has fewer inequalities to check, each inequality involves the change in value when adding only 2 elements to a set.

These properties seem fairly similar in that they involve inequalities of the set derivatives. However, there are several other characterizations, some of which appear to have nothing to do with second order differences being negative. For example, a set function f is submodular if and only if for *all* $\mathbf{c} \in \mathbb{R}^n$ it holds that $A, B \in \mathcal{Q}_{\mathbf{c}} \Rightarrow A \cap B,\ A \cup B \in \mathcal{Q}_{\mathbf{c}}$, where $\mathcal{Q}_{\mathbf{c}} := \arg\min_A f(A) + \langle \mathbb{1}_A, \mathbf{c} \rangle$. In words, this means that the collection of minimizers of f plus any modular function is a lattice. See [Fuj05] for a proof. Another example of a rather striking necessary and sufficient condition for submodularity is given in Proposition 3.4.

Maximization While the exact maximization of general submodular functions is a hard problem, maximizing a nonnegative nondecreasing submodular function under a cardinality constraint can be done to within a constant factor of the optimum. That is, if one wishes to find a maximal set of size k, then a simple greedy algorithm[2] will give a set of cardinality k with function value no less than $1-1/e$ of the maximal set of cardinality k [NWF78]. This useful property of approximate maximization is is a major driving force behind the interest in submodular functions, as it has many practical applications and useful generalizations. However, aside from our experiments showcased in Figure 6.1, we otherwise focus on the problem of submodular minimization.

3.3.1 Convex Analysis of Submodularity

We give an overview of how convex analysis can be used to analyze submodular functions. While this presentation is original, the material is standard [Sch03, Fuj05, Bac11].

We start by defining some important polyhedra used in the analysis of submodular functions. We can define the subdifferential of a set function in a form analogous to continous functions, except modular set functions play the role of linear functionals. Any vector $\boldsymbol{\lambda} \in \mathbb{R}^n$ defines a modular function on E, which we can denote in different ways, depending on what is convenient for the context:

$$2^E \ni A \mapsto \boldsymbol{\lambda}(A) := \langle \mathbb{1}_A, \boldsymbol{\lambda} \rangle = \sum_{a \in A} \lambda[a]$$

Then the subdifferential of a set function at a set is defined as a modular function which lower bounds the change in value of the function relative to that set. That is, $\boldsymbol{\lambda}$ is subgradient for f at the set A, if for all sets B the difference $f(B)-f(A)$ is bounded below by difference $\boldsymbol{\lambda}(B)-\boldsymbol{\lambda}(A)$.

$$\partial f(A) := \{\, \boldsymbol{\lambda} \in \mathbb{R}^n \mid f(B) \geq f(A) + \langle \mathbb{1}_B - \mathbb{1}_A, \boldsymbol{\lambda} \rangle \text{ for all } B \in 2^E \,\}$$

We also refer to this as the discrete subdifferential. Unlike a continuous subdifferential, which may be empty if a function is nonconvex, the discrete subdifferential is always nonempty. In fact, the subdifferential for the set A is unbounded along every ray in the direction $\mathbb{1}_A - \mathbb{1}_{E\setminus A}$. This is true for *any* set function. This is because $\partial f(A)$ is a polyhedron defined by the normal directions $\mathbb{1}_B - \mathbb{1}_A$ where $B \in 2^E \setminus \{A\}$, and for any such direction $\langle \mathbb{1}_B - \mathbb{1}_A, \mathbb{1}_A - \mathbb{1}_{E\setminus A} \rangle =$

[2]Add elements one at a time by choosing whichever element increases the function the most.

$-|A \ominus B| < 0$. So moving far enough in the direction $\mathbb{1}_A - \mathbb{1}_{E \setminus A}$ will always result in a subgradient. Precisely, for any $\boldsymbol{\lambda} \in \mathbb{R}^n$, we have:

$$\boldsymbol{\lambda} + t(\mathbb{1}_A - \mathbb{1}_{E\setminus A}) \in \partial f(A) \quad \Longleftrightarrow \quad t \geq \max_{\substack{B \in 2^E \\ B \neq A}} \frac{f(A) - f(B) + \boldsymbol{\lambda}(B) - \boldsymbol{\lambda}(A)}{|A \ominus B|}$$

In what follows, we will assume that the set function f is 'normalized', it has $f(\emptyset) = 0$, possibly by subtracting an offset. The submodular polyhedron P_f and base polytope B_f are thus defined by:

$$P_f := \{ \boldsymbol{\lambda} \in \mathbb{R}^n \mid \langle \mathbb{1}_A, \boldsymbol{\lambda} \rangle \leq f(A) \text{ for all } A \in 2^E \} \tag{3.3.1}$$

$$B_f := \{ \boldsymbol{\lambda} \in \mathbb{R}^n \mid \langle \mathbb{1}, \boldsymbol{\lambda} \rangle = f(E), \text{ and } \langle \mathbb{1}_A, \boldsymbol{\lambda} \rangle \leq f(A) \text{ for all } A \in 2^E \} \tag{3.3.2}$$

It is easy to see that that $P_f = \partial f(\emptyset)$. Furthermore,

$$P_f \cap \partial f(A) = P_f \cap \{ \boldsymbol{\lambda} \in \mathbb{R}^n \mid \boldsymbol{\lambda}(A) = f(A) \}$$

In particular, $B_f = \partial f(\emptyset) \cap \partial f(E)$. Note that all vectors in the set B_f must satisfy $\lambda[a] \leq f(a)$ and $\langle \mathbb{1}, \boldsymbol{\lambda} \rangle = f(E)$, and thus $\lambda[b] \geq f(E) - \sum_{a \in E-b} f(a)$. So the set B_f is bounded and we are justified in referring to it as a polytope. The elements of the base polytope are called the bases of the function. Note when f is the rank function of a matroid (cf. Appendix A), each extreme point of the base polytope corresponds to the indicator vector of a base (a.k.a. basis) for the matroid.

The base polytope essentially contains all the subgradients needed to fully characterize a submodular function. While it is defined by exponentially many constraints, there is a simple formula to optimize a linear function over it if we have oracle access to the function, if and only if the function is submodular. The key insight due to Edmonds [Edm70] is that solving $\max_{\boldsymbol{\lambda} \in B_f} \langle \mathbf{x}, \boldsymbol{\lambda} \rangle$ depends only on the ordering of the components of $\mathbf{x}$.

We say that a permutation $\pi \in S_n$ (S_n is the nth symmetric group, the set of all bijections from E to itself) is consistent with a vector $\mathbf{x}$, if the components of $\mathbf{x}$ are nonincreasing with respect to that permutation. Let K_π be the cone of all vectors consistent with π, and K_π° be its polar cone.

$$K_\pi := \{\, \mathbf{x} \in \mathbb{R}^n \mid x[\pi_1] \geq x[\pi_2] \ldots \geq x[\pi_n] \,\} \tag{3.3.3}$$

$$K_\pi^\circ := \{\, \boldsymbol{\lambda} \in \mathbb{R}^n \mid \langle \mathbb{1}, \boldsymbol{\lambda} \rangle = 0, \text{ and } \textstyle\sum_{i=1}^k \lambda[\pi_i] \leq 0 \text{ for } k = 1 \ldots n \,\} \tag{3.3.4}$$

Note that these cones are proper with respect to the subspace of vectors orthogonal to $\mathbb{1}$.

Given a permutation π, consider a sequence of sets starting with the empty set, adding one element at a time in the order specified by π. Then for a set function f, define $\boldsymbol{v}_{\pi,f} \in \mathbb{R}^n$ to be the vector with coordinates equal to the changes in value of f over this sequence of sets. For this, it will be useful to define ϖ^k to be the subset of E consisting of the first k elements in π.

$$\boldsymbol{v}_{\pi,f}[\pi_k] := \Delta_{\pi_k} f(\varpi^{k-1}) = f(\varpi^k) - f(\varpi^{k-1}) \tag{3.3.5}$$

$$\varpi^k := \{\pi_1, \ldots, \pi_k\} \tag{3.3.6}$$

The points $\boldsymbol{v}_{\pi,f}$ are called the extreme bases of the function f; we will justify the name by proving that they are indeed the vertices of the base polytope. First, we need to prove that they are even in the base polytope.

Proposition 3.4. *The set function f is submodular if and only if $\boldsymbol{v}_{\pi,f} \in B_f$ for all $\pi \in S_n$.*

Proof. Suppose f is not submodular. Then $f(A+b+c) + f(A) > f(A+b) + f(A+c)$ for some $b, c \in E$, $A \subseteq E - b - c$. Let π be a permutation with $\varpi^k = A, A+b, A+b+c$ for $k = |A|, |A|+1, |A|+2$ respectively. This means $\langle \mathbb{1}_A, \boldsymbol{v}_{\pi,f} \rangle = f(A)$, $\boldsymbol{v}_{\pi,f}[c] = f(A+b+c) - f(A+b)$ and thus:

$$\langle \mathbb{1}_{A+c}, \boldsymbol{v}_{\pi,f} \rangle = \langle \mathbb{1}_A, \boldsymbol{v}_{\pi,f} \rangle + \langle \mathbb{1}_c, \boldsymbol{v}_{\pi,f} \rangle = f(A) + f(A+b+c) - f(A+b) > f(A+c)$$

So we conclude $\boldsymbol{v}_{\pi,f}$ violates a constraint and is not in the polytope B_f.

Otherwise, suppose f is submodular. Clearly $\langle \mathbb{1}, \boldsymbol{v}_{\pi,f} \rangle = f(E)$ for all permutations π, so the equality constraint in the definition of B_f is satisfied. Let A be any set and π be any permutation. We wish to show that $\langle \mathbb{1}_A, \boldsymbol{v}_{\pi,f} \rangle \leq f(A)$. Let the increasing sequence $k(\cdot)$ correspond to the indices of A relative to the permutation π. That is, $A = \{\pi_{k(1)}, \pi_{k(2)}, \ldots, \pi_{k(|A|)}\}$.

This means we have for $j = 1 \ldots |A|$, $A \cap \varpi^{k(j)-1} = \{\pi_{k(1)}, \pi_{k(2)}, \ldots, \pi_{k(j-1)}\} = A \cap \varpi^{k(j-1)}$.

$$\begin{aligned}
\langle \mathbb{1}_A, \boldsymbol{\nu}_{\pi,f} \rangle &= \sum_{j=1}^{|A|} f(\varpi^{k(j)}) - f(\varpi^{k(j)-1}) \\
&\le \sum_{j=1}^{|A|} f(A \cap \varpi^{k(j)}) - f(A \cap \varpi^{k(j)-1}) && \text{(submodularity)} \\
&= \sum_{j=1}^{|A|} f(A \cap \varpi^{k(j)}) - f(A \cap \varpi^{k(j-1)}) && \text{(definition of } k(\cdot)) \\
&= f(A \cap \varpi^{k(|A|)}) = f(A)
\end{aligned}$$

So we conclude $\langle \mathbb{1}_A, \boldsymbol{\nu}_{\pi,f} \rangle \le f(A)$ for all $A \in 2^E$ so $\boldsymbol{\nu}_{\pi,f} \in B_f$. □

Note that Proposition 3.4 gives yet another characterization of submodularity. For the rest of this section, we will assume that f is submodular. Having established that the points defined by Equation 3.3.5 are bases, we can characterize them further:

Proposition 3.5. *The point $\boldsymbol{\nu}_{\pi,f}$ is the unique element of B_f which maximizes the inner product with any element of K_π.*

$$\bigcap_{\mathbf{x} \in K_\pi} \arg\max_{\boldsymbol{\lambda} \in B_f} \langle \mathbf{x}, \boldsymbol{\lambda} \rangle = \left\{ \boldsymbol{\nu}_{\pi,f} \right\}$$

Furthermore, $\boldsymbol{\nu}_{\pi,f}$ is extreme in B_f.

Proof. First we show that $\boldsymbol{\nu}_{\pi,f}$ is maximal when $\mathbf{x}$ is a corner of the unit cube. Let A be any set such that $\mathbb{1}_A \in K_\pi$. That is, $A = \varpi^k$ for $k = |A|$. Note that by definition of B_f, the maximal value $\max_{\boldsymbol{\lambda} \in B_f} \langle \mathbb{1}_A, \boldsymbol{\lambda} \rangle$ must be no more than $f(A)$. By construction, $\langle \mathbb{1}_{\varpi^k}, \boldsymbol{\nu}_{\pi,f} \rangle = \sum_{j=1}^{k} f(\varpi^j) - f(\varpi^{j-1}) = f(\varpi^k) = f(A)$, so $\boldsymbol{\nu}_{\pi,f}$ achieves this maximum value. By Proposition 3.4, $\boldsymbol{\nu}_{\pi,f}$ is an element of B_f. So we conclude $\boldsymbol{\nu}_{\pi,f} \in \arg\max_{\boldsymbol{\lambda} \in B_f} \langle \mathbb{1}_{\varpi^k}, \boldsymbol{\lambda} \rangle$ for $k = 1 \ldots n$.

Next, suppose $\mathbf{x}$ is an arbitrary point in K_π. Then we can decompose it into a sum: $\mathbf{x} = x[\pi_n]\mathbb{1} + \sum_{k=1}^{n-1} \alpha_k \mathbb{1}_{\varpi^k}$ with nonnegative coefficients $\alpha_k = x[\pi_k] - x[\pi_{k+1}] \ge 0$. Since $\langle \mathbb{1}, \boldsymbol{\lambda} \rangle$ is constant over B_f, and $\boldsymbol{\nu}_{\pi,f}$ is maximal for each term in the sum, we conclude that $\boldsymbol{\nu}_{\pi,f} \in \arg\max_{\boldsymbol{\lambda} \in B_f} \langle \mathbf{x}, \boldsymbol{\lambda} \rangle$.

To show uniqueness, suppose $\boldsymbol{\lambda}, \boldsymbol{\lambda}' \in B_f$ both maximize the inner product for all $\mathbf{x} \in K_\pi$. Then $\langle \mathbf{x}, \boldsymbol{\lambda} \rangle \le \langle \mathbf{x}, \boldsymbol{\lambda}' \rangle$ for all $\mathbf{x} \in K_\pi$, which is equivalent to $\boldsymbol{\lambda} - \boldsymbol{\lambda}' \in K_\pi^\circ$. Likewise, $\boldsymbol{\lambda}' - \boldsymbol{\lambda} \in K_\pi^\circ$, but since K_π° is pointed, this implies that $\boldsymbol{\lambda} = \boldsymbol{\lambda}'$.

Similarly, to see that $\boldsymbol{\nu}_{\pi,f}$ is extreme in B_f, suppose that $\boldsymbol{\nu}_{\pi,f} = \theta\boldsymbol{\lambda}_1 + (1-\theta)\boldsymbol{\lambda}_2$ for some $\boldsymbol{\lambda}_1, \boldsymbol{\lambda}_2 \in B_f$, with $\theta \in (0,1)$. The optimality of $\boldsymbol{\nu}_{\pi,f}$ implies that $\langle \mathbf{x}, \boldsymbol{\lambda}_2 \rangle \le \langle \mathbf{x}, \boldsymbol{\nu}_{\pi,f} \rangle$ for all $\mathbf{x} \in K_\pi$, which is equivalent to $\boldsymbol{\lambda}_2 - \boldsymbol{\nu}_{\pi,f} = \theta(\boldsymbol{\lambda}_1 - \boldsymbol{\lambda}_2) \in K_\pi^\circ$. Likewise $\boldsymbol{\lambda}_1 - \boldsymbol{\nu}_{\pi,f} = (1-\theta)(\boldsymbol{\lambda}_2 - \boldsymbol{\lambda}_1) \in K_\pi^\circ$. Again by using the fact that K_π° is a pointed cone, we have $\boldsymbol{\lambda}_2 - \boldsymbol{\lambda}_1 = 0$, and thus $\boldsymbol{\nu}_{\pi,f} = \boldsymbol{\lambda}_1 = \boldsymbol{\lambda}_2$. □

So we have justified the name extreme base for the vectors $\boldsymbol{\nu}_{\pi,f}$. In fact, these are the *only* extreme elements of B_f. That is, we can characterize the base polytope as the convex hull of the collection $\{\boldsymbol{\nu}_{\pi,f}\}_{\pi \in S_n}$.

Proposition 3.6. $B_f = \operatorname{conv}\{\boldsymbol{\nu}_{\pi,f}\}_{\pi \in S_n}$.

Proof. Clearly every $\mathbf{x} \in \mathbb{R}^n$ is contained in some cone K_π. Thus Proposition 3.5 implies that $\max_{\boldsymbol{\lambda} \in B_f} \langle \mathbf{x}, \boldsymbol{\lambda} \rangle = \max_{\pi \in S_n} \langle \mathbf{x}, \boldsymbol{\nu}_{\pi,f} \rangle$ for all $\mathbf{x} \in \mathbb{R}^n$. So if $C = \operatorname{conv}\{\boldsymbol{\nu}_{\pi,f}\}_{\pi \in S_n}$, we have $\sigma_{B_f} = \sigma_C$ everywhere. Both sets are closed and convex, thus $\delta_{B_f} = \sigma_{B_f}^* = \sigma_C^* = \delta_C$, so we conclude they are identical. □

3.3.2 Lovász Extension

Let us review what we have established about σ_{B_f}, the support function of the base polytope of f. First, what does it evaluate to at a corner of the unit cube? By Proposition 3.4, the points $\boldsymbol{\nu}_{\pi,f}$ are contained in the base polytope, therefore for any set $A \in 2^E$, the inequality $\langle \mathbb{1}_A, \boldsymbol{\lambda} \rangle \le f(A)$ is satisfied with equality for some $\boldsymbol{\lambda} \in B_f$ (namely any $\boldsymbol{\nu}_{\pi,f}$ for which $\mathbb{1}_A \in K_\pi$). Thus $\sigma_{B_f}(\mathbb{1}_A) = \max_{\boldsymbol{\lambda} \in B_f} \langle \mathbb{1}_A, \boldsymbol{\lambda} \rangle = f(A)$. So therefore σ_{B_f} is an interpolation of f to the vertices of the unit cube. Furthermore, since it is a support function, it is convex by construction, making it a natural tool for use in minimization. Since this function is so important, we will denote it simply $\tilde{f}$, and we refer to it as the Lovász extension of f.

$$\tilde{f}(\mathbf{x}) := \sigma_{B_f}(\mathbf{x}) = \max_{\boldsymbol{\lambda} \in B_f} \langle \mathbf{x}, \boldsymbol{\lambda} \rangle$$

On a historical note, this is named after Lovász, who was the first to consider it as a tool for minimization [GLS81, Lov83]. It appeared earlier in the work of Edmonds [Edm70] implicitly, in that he showed how maximizing a linear function over the base polytope can be done in a greedy fashion.

Since B_f is bounded, $\tilde{f}$ has full domain. The subgradient of the Lovász extension at a point $\mathbf{x}$ is the convex hull of all base vertices for permutations consistent with that vector:

$$\partial \tilde{f}(\mathbf{x}) = \operatorname{conv}\{\, \boldsymbol{\nu}_{\pi,f} \mid \mathbf{x} \in K_\pi \,\} \tag{3.3.7}$$

In particular, the subgradients at a corner of the unit cube are also discrete subgradients for f.

$$\partial f(A) \cap B_f = \operatorname{conv}\{\, \boldsymbol{\nu}_{\pi,f} \mid A = \{\pi_1, \ldots, \pi_{|A|}\}\,\} \tag{3.3.8}$$

Consider what Equation 3.3.7 implies about the relationship between subdifferentials at different points. If the point $\mathbf{y}$ is consistent with at least the permutations with which $\mathbf{x}$ is consistent, then all subgradients at $\mathbf{x}$ are also subgradients at $\mathbf{y}$:

$$\mathbf{x} \in K_\pi \text{ for all } \pi \text{ such that } \mathbf{y} \in K_\pi \Rightarrow \partial \tilde{f}(\mathbf{x}) \subseteq \partial \tilde{f}(\mathbf{y})$$

In particular, by relating the subdifferential of $\mathbf{x}$ to the corner point $\mathbb{1}_A$, we can derive a condition for when the continous subgradients for $\tilde{f}$ are also discrete subgradients for f. Given a point $\mathbf{x}$, defining X_α^+ to be the subset of components of $\mathbf{x}$ greater than or equal to α, it is not hard to see that $\mathbf{x} \in K_\pi$ if and only if $\mathbb{1}_{X_\alpha^+} \in K_\pi$ for all $\alpha \in \mathbb{R}$. So by Equation 3.3.7, we get:

$$\begin{aligned} X_\alpha^+ &:= \{\, e \in E \mid x[e] \geq \alpha \,\} \\ \partial \tilde{f}(\mathbf{x}) &= \bigcap_{\alpha \in \mathbb{R}} \partial f(X_\alpha^+) \cap B_f \end{aligned} \tag{3.3.9}$$

Note that there is a subtlety to this characterization that is important for algorithms that use the Lovász extension to minimize the set function. In order to conclude that the continuous subdifferential $\partial \tilde{f}(\mathbf{x})$ is included in the set function subdifferential $\partial f(A)$, it is necessary that the components of $\mathbf{x}$ in A be *strictly* separated from the complementary components.

$$\min_{a \in A} x[a] > \max_{b \in E \setminus A} x[b] \quad \Rightarrow \quad \partial \tilde{f}(\mathbf{x}) \subseteq \partial f(A) \tag{3.3.10}$$

$$\min_{a \in A} x[a] \geq \max_{b \in E \setminus A} x[b] \quad \not\Rightarrow \quad \partial \tilde{f}(\mathbf{x}) \subseteq \partial f(A) \tag{3.3.11}$$

Note that the premise of Equation 3.3.11 does imply that $\partial \tilde{f}(\mathbf{x}) \cap \partial f(A)$ is nonempty.

3.3.3 Examples of Base Polytopes

Which is more important—a submodular function or its base polytope? Of course, given the dual relationship between the two, one cannot really be more important than the other, but historically, base polytopes were discovered first by Edmonds. Any base polytope is the union of the faces of a polymatroid. (Interestingly, polymatroids can be defined without mentioning submodularity.)

We start with a short proof of an important property of submodular functions. For any submodular function, minimizing out a subset of the variables results in another submodular function. This is exactly analogous to convex functions. Suppose a the function $g(A,B)$ is a submodular function on $2^{E\times F}$, where $A \in 2^E$, $B \in 2^F$. Then suppose f is given by minimizing g over all arguments B:

$$f(A) = \min_{B\subseteq F} g(A,B) \tag{3.3.12}$$

It is simple to show if g is submodular, then f must be as well:

$$\begin{aligned}
f(A_1) + f(A_2) &= g(A_1,B_1) + g(A_2,B_2) && \text{Where } B_i = \arg\min_{B\subseteq F} g(A_i,B) \\
&\geq g(A_1\cup A_2, B_1\cup B_2) + g(A_1\cap A_2, B_1\cap B_2) && \text{(Submodularity of } g\text{)} \\
&\geq f(A_1\cup A_2) + f(A_1\cap A_2) && \text{(Minimality of } f\text{)}
\end{aligned}$$

In particular, when a function f is *graph representable*, it can be written as a minimum over submodular $g(A,B)$, where g is a fully general quadratic function.

$$f(A) = \mathbf{c}(A) + \min_B \mathbf{s}(B) + \mathbb{1}_A^\top \mathbf{W}_{11}\mathbb{1}_{E\setminus A} + \mathbb{1}_A^\top \mathbf{W}_{12}\mathbb{1}_{F\setminus B} + \mathbb{1}_B^\top \mathbf{W}_{22}\mathbb{1}_{F\setminus B}$$

The term graph representable refers to the fact that a second-order submodular function is a cut function of a graph. We can express points in the base polytope of f through $2^{|E|+|F|}$ linear inequalities. That is, $\boldsymbol{\lambda} \in B_f$ if:

$$\begin{gathered}
-\mathbb{1}_A^\top(\boldsymbol{\lambda} - \mathbf{c}) + \mathbf{s}^\top\mathbb{1}_B + \mathbb{1}_A^\top \mathbf{W}_{11}\mathbb{1}_{E\setminus A} + \mathbb{1}_A^\top \mathbf{W}_{12}\mathbb{1}_{F\setminus B} + \mathbb{1}_B^\top \mathbf{W}_{22}\mathbb{1}_{F\setminus B} \geq 0 \\
\text{for all } A \in 2^E, B \in 2^F.
\end{gathered}$$

Now consider a simpler case where all entries of $\mathbf{W}_{11}$ and $\mathbf{W}_{22}$ equal zero. This means the graph is bipartite: one partition for E and and the other for the variables to be minimized over. Such a function can then be expressed as:

$$\begin{aligned} f(A) &= \mathbb{1}_A^\top \mathbf{c} + \min_B \mathbb{1}_A^\top \mathbf{W}(\mathbb{1} - \mathbb{1}_B) + \mathbf{s}^\top \mathbb{1}_B \\ &= \mathbf{c}(A) + \sum_j \min(s_j, \mathbf{w}_j(A)) \end{aligned}$$

For this function, we can introduce new variables to give a simple description of the base polytope. That is, $\boldsymbol{\lambda} \in B_f$ if there exists some $\mathbf{Y} \in \mathbb{R}^{|E|\times|F|}$ such that:

$$\boldsymbol{\lambda} = \mathbf{c} + \mathbf{Y}^\top \mathbb{1}, \qquad \mathbf{0} \le \mathbf{Y} \le \mathbf{W}, \qquad \mathbf{Y}\mathbb{1} = \mathbf{s}.$$

So testing membership in this base polytope is much simpler than the general case. What happens if we iterate Equation 3.3.12, by introducing a new bipartite graph between sets of variables. If we do this k times, we get a graph representable function, only now the graph is a trellis with k levels:

$$f(A) = \mathbb{1}_A^\top \mathbf{c} + \min_{B_1,\dots B_k} \mathbb{1}_A^\top \mathbf{W}_1(\mathbb{1} - \mathbb{1}_{B_1}) + \sum_{j=2}^{k} \mathbb{1}_{B_{j-1}}^\top \mathbf{W}_j(\mathbb{1} - \mathbb{1}_{B_j}) + \mathbf{s}^\top \mathbb{1}_{B_k} \tag{3.3.13}$$

Again, this function has an efficiently representable base polytope. That is, we have $\boldsymbol{\lambda} \in B_f$ if for some matrices $\mathbf{Y}_1, \dots, \mathbf{Y}_k$, we have:

$$\begin{aligned} \boldsymbol{\lambda} &= \mathbf{c} + \mathbf{Y}_1^\top \mathbb{1}, & \mathbf{0} &\le \mathbf{Y}_1 \le \mathbf{W}_1, \\ \mathbf{Y}_j \mathbb{1} &= \mathbf{Y}_{j+1}^\top \mathbb{1}, & \mathbf{0} &\le \mathbf{Y}_{j+1} \le \mathbf{W}_{j+1} \text{ for } j = 1 \dots k-1, \\ \mathbf{Y}_k \mathbb{1} &= \mathbf{s}. \end{aligned} \tag{3.3.14}$$

3.4 Submodular Minimization

The general problem we are interested in is finding a global minimizer of a submodular function. Finding Before discussing specific algorithmic techniques, let us review what the optimality conditions and duality gap are for solving submodular minimization as a convex minimization problem. Recall that $\sigma_{[0,1]^n}(\mathbf{x}) = \max_{\boldsymbol{\lambda} \in [0,1]^n} \langle \mathbf{x}, \boldsymbol{\lambda} \rangle = \sum_{e \in E} \max(x[e], 0)$ is the support function of the unit cube.

Theorem 3.7 (Submodular Minimization Optimality Conditions). *We have for all $A \in 2^E$, $\boldsymbol{\lambda} \in B_f$:*

$$f(A) + \sigma_{[0,1]^n}(-\boldsymbol{\lambda}) \geq 0 \tag{3.4.1}$$

This equation is satisfied with equality if and only if $(\mathbb{1}_A - \mathbb{1}_{E\setminus A}) \bullet \boldsymbol{\lambda} \leq \mathbf{0}$[3]*, and $\boldsymbol{\lambda} \in \partial f(A)$. Furthermore this is true if and only if A minimizes f over 2^E, and $\boldsymbol{\lambda}$ maximizes $-\sigma_{[0,1]^n}(-\boldsymbol{\lambda})$ over B_f.*

Proof. For any $A \in 2^E$, $\boldsymbol{\lambda} \in B_f$:

$$\begin{aligned}
-\sigma_{[0,1]^n}(-\boldsymbol{\lambda}) &= \min_{\mathbf{x}\in[0,1]^n} \langle \mathbf{x}, \boldsymbol{\lambda}\rangle && \text{(def. of } \sigma_{[0,1]^n}) && (3.4.2)\\
&\leq \langle \mathbb{1}_A, \boldsymbol{\lambda}\rangle && \text{(minimality of } \mathbf{x}) && (3.4.3)\\
&\leq f(A) && \text{(def. of } B_f) && (3.4.4)
\end{aligned}$$

This demonstrates Equation 3.4.1. If $(\mathbb{1}_A - \mathbb{1}_{E\setminus A}) \bullet \boldsymbol{\lambda} \leq \mathbf{0}$, then $\lambda[a] \leq 0$ for all $a \in A$ and $\lambda[b] \geq 0$ for all $b \in E\setminus A$, so then $\mathbb{1}_A$ is an optimal $\mathbf{x}$ in Equation 3.4.2, and thus Equation 3.4.3 is an equality. Conversely, if $\lambda[a] > 0$ for some $a \in A$ or $\lambda[b] < 0$ for some $b \in E \setminus A$, then Equation 3.4.3 must be a strict inequality. Finally, a base vector $\boldsymbol{\lambda}$ is contained in $\partial f(A)$ if and only if $\langle \mathbb{1}_A, \boldsymbol{\lambda}\rangle = f(A)$, i.e. Equation 3.4.4 is satisfied with equality. Finally, note that since the bound in Equation 3.4.1 is universal, it can only hold with equality for optimal A and $\boldsymbol{\lambda}$. This establishes conditions for when equality can hold, but we need to demonstrate that it always will. That is, Equation 3.4.1 is equivalent to the following statement of weak duality, but we additionally need strong duality:

$$\max_{\boldsymbol{\lambda}\in B_f} \min_{\mathbf{x}\in[0,1]^n} \langle \mathbf{x}, \boldsymbol{\lambda}\rangle \leq \min_{\mathbf{x}\in[0,1]^n} \max_{\boldsymbol{\lambda}\in B_f} \langle \mathbf{x}, \boldsymbol{\lambda}\rangle \tag{3.4.5}$$

Since the constraints are polyhedral, strong duality holds by Fenchel's Duality Theorem. □

3.4.1 Ellipsoid Method and Polynomial Time Algorithms

The ellipsoid method has been of historical importance in that it can be used to establish the possibility of polynomial time optimization of general convex functions. For example, Hačijan used it in the late 1970s to prove that linear programs were polynomial time solvable [Hač79].

[3] We use the symbol $\bullet$ to denote the elementwise or Hadamard product of vectors.

While its generality is a great strength for proving theorems, in practice the problems it can be applied to have specialized methods that work much faster.

Suppose we wish to minimize a convex function using an oracle which takes as input any point in the function's domain and returns the function value and a subgradient. We can use the fact that for any current iterate $\mathbf{x}^k$ with subgradient $\boldsymbol{\lambda}^k \in \partial f(\mathbf{x}^k)$, a minimizer $\mathbf{x}^* \in \arg\min f$ must be contained in the half-space $\langle \mathbf{x}^*, \boldsymbol{\lambda}^k \rangle \leq \langle \mathbf{x}^k, \boldsymbol{\lambda}^k \rangle + f^k_{\text{best}} - f(\mathbf{x}^k)$, where f^k_{best} is the smallest value of f encountered up to step k. Splitting algorithms are a general class of algorithms that maintain a feasible region such as a polytope or an ellipsoid, then use oracle information to splits the feasible region and reduce it somehow.

In particular, a simple variant of the ellipsoid algorithm might proceed as follows. Start with ellipsoid $\mathcal{E}^0$ large enough to be guaranteed to contain a minimizer of f, set $k = 0$ and begin: Let $\mathbf{x}^k$ be the the centroid of $\mathcal{E}^k$, and use the oracle to get a subgradient $\boldsymbol{\lambda}^k \in \partial f(\mathbf{x}^k)$. This gives a half-space that splits the current ellipsoid, namely $H^k := \{\mathbf{x} \in \mathbb{R}^n \mid \langle \mathbf{x}, \boldsymbol{\lambda}^k \rangle \leq \langle \mathbf{x}^k, \boldsymbol{\lambda}^k \rangle + f^k_{\text{best}} - f(\mathbf{x}^k)\}$. Let $\mathcal{E}^{k+1}$ be the ellipsoid of minimal *volume* which contains the intersection $H^k \cap \mathcal{E}^k$. Increment k and repeat these iterations until an iterate is sufficiently close to optimality.

To represent ellipsoids numerically, one can use a positive definite matrix and a vector. Under this representation, computing the minimal volume ellipsoid for the next iterate amounts to a rank-one update rule of the matrix. The algorithm can be analyzed by bounding the rate at which the volume of these ellipsoids decrease.

The Lovász extension $\tilde{f}$ is a convex function, and by Equation 3.3.7, a subgradient at any point can be generated efficiently, provided evaluation of the set function f is efficient. So one can use a generic convex minimization algorithm such as an ellipsoid method to minimize $\tilde{f}$ over the unit cube. Note that the original ellipsoid method proposed for submodular minimization in [GLS81] differs somewhat from the algorithm outlined above, but the important ideas are the same: using the convexity of $\tilde{f}$ and an efficient subgradient oracle to minimize the set function.

However, the asymptotic running time of the ellipsoid method depends on the values of the function itself, not just the size of the problem. Hence its running time is actually weakly polynomial. Since then, several strongly polynomial algorithms for submodular minimization have been developed. [IFF01] [IO09] Unfortunately, these are all of complexity at least $O(n^5)$, and they are not empirically fast enough to be of practical use for all but very small problems.

3.4.2 Fujishige's Minimal Norm Algorithm

As originally shown by Fujishige [FI11], finding the point of the base polytope B_f of minimal ℓ_2 norm is equivalent to solving the submodular minimization problem. (We prove a more general version of this equivalance in Theorem 5.4.)With access to an oracle that maximizes a linear function over a basepolytope, one uses the Frank-Wolfe algorithm to find the minimum norm point in that polytope. The algorithm is guaranteed to converge in finite time because there are only finitely many possible subsets of vertices, and it is a descent algorithm which never returns to the same state. In general the algorithm could require exponentially many calls to an oracle, albeit no problem instance exhibiting this behavior is known.

To satisfy the role of the oracle, we need an an efficient subroutine which can solve $\arg\max_{\boldsymbol{\lambda}\in B_f} \langle \mathbf{x}, \boldsymbol{\lambda} \rangle$ for any $\mathbf{x} \in \mathbb{R}^n$. By Proposition 3.5, we know that if $\mathbf{x} \in K_\pi$, the point $\boldsymbol{\nu}_{\pi,f}$ is a maximizing vertex of the polytope for that functional. So, we just need to sort[4] the components of $\mathbf{x}$ to find a permutation with which $\mathbf{x}$ is consistent, and then our oracle returns the base vertex given by the formula in Equation 3.3.5. This procedure is known as the greedy algorithm, because if f is the rank function of a matroid, it is equivalent to adding elements to an independent set greedily depending on their weight $x[e]$.

Despite there being no subexponential upper bounds on the running time, this algorithm works quite well in practice. The exact performance will depend some subtleties, such as how expensive it is to compute the submodular function. If f is a complicated function such as a log-determinant, then computing $\boldsymbol{\nu}_{\pi,f}$ given π may actually be the bottleneck of the program.

3.4.3 Special Cases

There are certain special cases of submodular minimization for which faster algorithms are known to exist. Even these are not necessarily useful in practice, but they are always asymptotically faster than the algorithms which make no assumptions other than submodularity.

[4]This can be a major source of nondeterminacy in an implementation if a sorting routine does not specify how ties are broken. Ties are not unlikely because the minimum norm base is expected to have many components of the same magnitude.

Symmetric Unconstrained minimization of a symmetric submodular function is trivial, since for such functions we have $2f(\emptyset) = f(\emptyset) + f(E) \leq f(A) + f(E \setminus A) \leq 2f(A)$ for all A, so $\emptyset$ and E trivially minimize A. However, if we restrict the search domain to exclude the extreme sets, the minimization problem is very much of interest. In this case, we can use an algorithm discovered by Queyranne [Que98] to minimize the a symmetric submodular function over the range $2^E \setminus \{\emptyset, E\}$.

Quadratic Potentials and Extensions The minimization of second-order submodular functions can be reformulated as finding the smallest cut of a graph with positive edge weights which separates two specific vertices. This is dual to finding the maximum flow between those two vertices, if the edge weights are capacities. This is a classic problem in computer science and there are a great variety of algorithms for solving it efficiently. One of the conceptually simplest of these is the Ford-Fulkerson algorithm which, in its simplest form, requires the weights to be integral.

Yet, even if a submodular function is not second order, it may still be possible to use min-cut/max-flow algorithms to minimize it. This is done by introducing extra Boolean variables to create a higher dimensional second order function, such that the original function is given by minimizing the second order function over the extra variables. Functions that admit such a representation are called graph representable. Much work has been done over the years to determine which functions are graph representable, and, furthermore, how to most efficiently represent the function, or at least approximate it efficiently. See for example, [KZ04], [JLB11]. Graph representable functions are necessarily submodular.

Chapter 4

Smoothed Gradient Methods for Decomposable Functions

4.1 Introduction

Several submodular minimization problems arising in machine learning and other domains have structure that allows for solving them more efficiently. Examples include symmetric functions that can be solved in $O(n^3)$ evaluations using Queyranne's algorithm [Que98], and functions that decompose into attractive, pairwise potentials, that can be solved using graph cutting techniques [FD05]. In this chapter, we introduce a novel class of submodular minimization problems that can be solved efficiently with convex optimization. In particular, we develop an algorithm SLG, that can minimize a class of submodular functions that we call *decomposable*. These are functions that can be decomposed into sums of concave functions applied to modular (additive) functions. Our algorithm is based on recent techniques of smoothed convex minimization [Nes05] applied to the Lovász extension. We demonstrate the usefulness of our algorithm on a joint classification-and-segmentation task involving tens of thousands of variables, and show that it outperforms state-of-the-art algorithms for general submodular function minimization by several orders of magnitude.

4.2 Background on Submodular Function Minimization

We are interested in minimizing set functions that map subsets of some base set E to real numbers; i.e. given $f : 2^E \to \mathbb{R}$ we wish to solve for $A^* \in \arg\min_A f(A)$. For simplicity of notation, we use the base set $E = \{1, \ldots n\}$, but in an application the base set may consist of nodes of a graph, pixels of an image, etc. Without loss of generality, we assume $f(\emptyset) = 0$. If the function f has no structure, then there is no way solve the problem other than checking all 2^n subsets. In this chapter, we consider functions that satisfy a key property that arises in many applications: *submodularity* (c.f., [Lov83]). A set function f is called submodular if and only if, for all $A, B \in 2^E$, we have

$$f(A \cup B) + f(A \cap B) \leq f(A) + f(B). \tag{4.2.1}$$

Submodular functions can alternatively, and perhaps more intuitively, be characterized in terms of their discrete derivatives. First, we define $\Delta_c f(A) = f(A \cup \{c\}) - f(A)$ to be the discrete derivative of f with respect to $c \in E$ at A; intuitively this is the change in f's value by adding the element c to the set A. Then, f is submodular if and only if:

$$\Delta_c f(A) \geq \Delta_c f(B), \text{ for all } A \subseteq B \subseteq E - c.$$

Note the analogy to concave functions; the discrete derivative is smaller for larger sets, in the same way that $\phi(x+h) - \phi(x) \geq \phi(y+h) - \phi(y)$ for all $x \leq y,\ h \geq 0$ if and only if ϕ is a concave function on $\mathbb{R}$. Thus a simple example of a submodular function is $f(A) = \phi(|A|)$ where ϕ is any concave function. Yet despite this connection to concavity, minimizing a submodular function can be done in polynomial time, wheras maximizing it is NP-hard. This is similar to the fact that it is easier to minimize a convex function than to maximize. One explanation for this is that submodular minimization can be reformulated as a convex minimization problem.

To see this, consider taking a set function minimization problem, and reformulating it as a minimization problem over the unit cube $[0,1]^n \subset \mathbb{R}^n$. Define $\mathbb{1}_A \in \mathbb{R}^n$ to be the indicator vector of the set A, i.e.,

$$\mathbb{1}_A[e] = \begin{cases} 0 \text{ if } e \notin A \\ 1 \text{ if } e \in A \end{cases}$$

We use the notation $x[e]$ for the component of the vector $\mathbf{x}$ corresponding to element e. Also we drop brackets and commas in subscripts, so $\mathbb{1}_{ab} = \mathbb{1}_{\{a,b\}}$ and $\mathbb{1}_a = \mathbb{1}_{\{a\}}$. For any vector $\boldsymbol{\lambda} \in \mathbb{R}^n$, we use the shorthand $\boldsymbol{\lambda}(A)$ for the corresponding modular function:

$$\boldsymbol{\lambda}(A) := \sum_{a \in A} \lambda[a] = \langle \mathbb{1}_A, \boldsymbol{\lambda} \rangle$$

A continuous extension of a set function is a function with a continuous argument that interpolates the values of the set function to the corners of the unit cube. That is, we should have $[0,1]^n \subseteq \operatorname{dom} \tilde{f} \subseteq \mathbb{R}^n$ and $f(A) = \tilde{f}(\mathbb{1}_A)$. Furthermore, we do not want the minimum of the continous function to be completely unrelated to the minimum of the set function. That is, if a particular set is a minimizer of the set function, then that should necessarily imply that the corresponding indicator vector is a minimizer of the continous extension over the unit cube:

$$A^* \in \underset{A \in 2^E}{\arg\min} f(A) \quad \Rightarrow \quad \mathbb{1}_{A^*} \in \underset{\mathbf{x} \in [0,1]^n}{\arg\min} \tilde{f}(\mathbf{x}). \tag{4.2.2}$$

A key result due to Lovász [Lov83] states that each submodular function f has an extension $\tilde{f}$ that not only satisfies the above property, but is also convex and efficient to evaluate. We can define the *Lovász extension* in terms of the base polytope B_f:

$$\tilde{f}(\mathbf{x}) = \max_{\boldsymbol{\lambda} \in B_f} \langle \boldsymbol{\lambda}, \mathbf{x} \rangle, \qquad B_f = \{ \boldsymbol{\lambda} \in \mathbb{R}^n \mid \boldsymbol{\lambda}(E) = f(E),\ \boldsymbol{\lambda}(A) \leq f(A),\ \text{for all } A \in 2^E \}.$$

The base polytope B_f is defined by exponentially many inequalities, and evaluating $\tilde{f}$ requires solving a linear program over this polyhedron. Perhaps surprisingly, as shown by Lovász, $\tilde{f}$ can be very efficiently computed as follows. For a fixed $\mathbf{x} \in \mathbb{R}^n$, let $\pi : E \to E$ be a permutation such that $x[\pi_1] \geq \ldots \geq x[\pi_n]$, and then define the set $\varpi^k = \{\pi_1, \ldots, \pi_k\}$. Then we have a formula for $\tilde{f}$ and a subgradient:

$$\tilde{f}(\mathbf{x}) = \sum_{k=1}^{n} x[\pi_k](f(\varpi^k) - f(\varpi^{k-1})), \quad \partial \tilde{f}(\mathbf{x}) \ni \sum_{k=1}^{n} \mathbb{1}_{\pi_k}(f(\varpi^k) - f(\varpi^{k-1})). \tag{4.2.3}$$

Note that if two components of $\mathbf{x}$ are equal, the above formula for $\tilde{f}$ is independent of the permutation chosen, but the subgradient is not unique.

Equation 4.2.2 was used to show that submodular minimization can be achieved in polynomial time [Lov83]. However, algorithms which directly minimize the Lovász extension are regarded as impractical. Despite being convex, the Lovász extension is non-smooth, and hence a simple subgradient descent algorithm would need $O(1/\epsilon^2)$ steps to achieve $O(\epsilon)$ accuracy.

Recently, Nesterov showed that if knowledge about the structure of a particular non-smooth convex function is available, it can be exploited to achieve a running time of $O(1/\epsilon)$

[Nes05]. One way this is done is to construct a smooth approximation of the non-smooth function, and then use an accelerated gradient descent algorithm which is highly effective for smooth functions. Connections of this work with submodularity and combinatorial optimization are also explored in [CN07] and [Bac10]. In fact, in [Bac10], Bach shows that computing the smoothed Lovász gradient of a general submodular function is equivalent to solving a submodular minimization problem. In this chapter, we do not treat general submodular functions, but rather a large class of submodular minimization functions that we call *decomposable*. (To apply the smoothing technique of [Nes05], special structural knowledge about the convex function is required, so it is natural that we would need special structural knowledge about the submodular function to leverage those results.) We further show that we can exploit the discrete structure of submodular minimization in a way that allows terminating the algorithm early with a certificate of optimality, which leads to dramatic performance improvements.

4.3 The Decomposable Submodular Minimization Problem

In this chapter, we consider the problem of minimizing submodular functions of the following form:

$$f(A) = \mathbf{c}(A) + \sum_j \phi_j(\mathbf{w}_j(A)), \tag{4.3.1}$$

where $\mathbf{c}, \mathbf{w}_j \in \mathbb{R}^n$ and $\mathbf{w}_j \geq \mathbf{0}$ and $\phi_j : [0, \mathbf{w}_j(E)] \to \mathbb{R}$ are arbitrary one-dimensional concave functions. Without loss of generality, we will assume, by rescaling if necessary, that $\|\mathbf{w}_j\|_\infty = 1$. It is shown in 4.4.1 that functions of this form are submodular. We call this class of functions *decomposable submodular functions*, as they decompose into a sum of concave functions applied to nonnegative modular functions. Below, we give examples of decomposable submodular functions arising in applications.

We first focus on the special case where all the concave functions ϕ_j are positive multiples of *threshold functions* $\min(\tau, \cdot)$ for some threshold $\tau > 0$. Since these functions are of key importance, we denote $\Psi(\tau, \mathbf{w}, A)$ to be the submodular function given by thresholding the modular function $\mathbf{w}(A)$ at threshold τ:

$$\Psi(\tau, \mathbf{w}, A) := \min(\tau, \mathbf{w}(A))$$

We refer to these submodular functions as *threshold potentials*. In Section 4.6, we will show how to generalize our approach to arbitrary decomposable submodular functions.

Examples. The simplest example is a *2-potential*, which has the form $\phi(|A \cap \{b,c\}|)$, where $\phi(1) - \phi(0) \geq \phi(1) - \phi(2)$. It can be expressed as a sum of a modular function and a threshold potential:

$$\phi(|A \cap \{b,c\}|) = \phi(0) + (\phi(2) - \phi(1))\langle \mathbb{1}_A, \mathbb{1}_{bc} \rangle + (2\phi(1) - \phi(0) - \phi(2))\Psi(1, \mathbb{1}_{bc}, A)$$

An example application where such function arises is in image classification schemes such as in [SWRC09]. In order to compute the Maximum a Posteriori configuration of a pairwise Markov Random Field model, one must minimize a sum of these potentials. On a high level, such an algorithm computes a value $p[b]$ that corresponds to the log-likelihood of pixel b being of one class vs. another, and for each pair of adjacent pixels, a value r_{bc} related to the log-likelihood that pixels b and c are of the same class. Then the algorithm classifies pixels by minimizing a sum of 2-potentials: $f(A) = \mathbf{c}(A) + \sum_{b,c} r_{bc}(1 - |1 - \langle \mathbb{1}_A, \mathbb{1}_{bc} \rangle|)$. If the value r_{bc} is large, this encourages the pixels b and c to be classified similarly.

More generally, consider a higher order potential function: a concave function of the number of elements in some activation set D, $\phi(|A \cap D|)$ where ϕ is concave. It can be shown that this can be written as a sum of a modular function and a positive linear combination of $|D| - 1$ threshold potentials with boolean weights ($\mathbf{w}_j \in \{0,1\}^n$). However, threshold potentials with nonuniform weights are strictly more general than those with boolean weights. That is, there exists τ and $\mathbf{w}$ such that $\Psi(\tau, \mathbf{w}, A)$ cannot be expressed as $\sum_j \phi_j(|D_j \cap A|)$ for *any* collection of concave ϕ_j and sets D_j.

Recent work [KLT09] has shown that classification performance can be improved by adding terms corresponding to such higher order potentials $\phi_j(|D_j \cap A|)$ to the objective function where the functions ϕ_j are piecewise linear concave functions, and the regions D_j of various sizes generated from a segmentation algorithm. Minimization of these particular potential functions can then be reformulated as a graph cut problem [KKT07], but this is less general than our approach.

Another canonical example of a submodular function is a set cover function. Such a function can be reformulated as a combination of threshold potentials with boolean weights and unit threshold (details in 4.4.2). Thus all functions that are weighted combinations of set cover functions can be expressed as threshold potentials.

Another example of decomposable functions arises in multiclass queuing systems [II07]. These are of the form $f(A) = \mathbf{u}(A)\phi(\mathbf{v}(A))$, where $\mathbf{u}, \mathbf{v}$ are nonnegative weight vectors and ϕ is a nonpositive nonincreasing concave function. We show in 4.4.3, that with the proper choice of ϕ_j and $\mathbf{w}_j$, this can in fact be reformulated as sum of the type in Equation 4.3.1 with n terms.

In our own experiments, shown in Section 4.7, we use an implementation of TextonBoost [SWRC09] and augment it with quadratic higher order potentials. That is, we use TextonBoost to generate per-pixel scores $\mathbf{c}$, and then minimize $f(A) = \mathbf{c}(A) + \sum_j |A \cap D_j||D_j \setminus A|$, where the regions D_j are regions of pixels that we expect to be of the same class (e.g., by running a cheap region-growing heuristic). The potential function $|A \cap D_j||D_j \setminus A|$ is smallest when A contains all of D_j or none of it. It gives the largest penalty when exactly half of D_j is contained in A. This encourages the classification scheme to classify most of the pixels in a region D_j the same way. We generate regions with a basic region-growing algorithm with random seeds. See Figure 4.1a for an illustration of examples of regions that we use. In our experience, this simple idea of using higher-order potentials can dramatically increase the quality of the classification over one using only 2-potentials, as can be seen in Figure 4.2.

4.4 Classification of Submodular Functions

In this section we explain why decomposable functions are submodular, describe some examples of decomposable functions, and show that decomposable functions are more general than some other classes of submodular functions that have been studied.

4.4.1 Submodularity of Decomposable Functions

Since the sum of submodular functions is submodular, we need only prove the submodularity of $f(A) = \phi(\mathbf{w}(A))$, where ϕ is an arbitrary one-dimensional concave function and $\mathbf{w} \geq \mathbf{0}$.

By definition of concavity, for all $\theta \in [0,1]$, we have:

$$\phi(\theta(y+h) + (1-\theta)x) + \phi((1-\theta)(y+h) + \theta x) \geq \phi(y+h) + \phi(x)$$

If $x \leq y$ and $h \geq 0$, then setting $\theta = h/(y - x + h)$ in the above gives us:

$$\phi(x+h) - \phi(x) \geq \phi(y+h) - \phi(y) \tag{4.4.1}$$

Then, for all $c \in E \setminus A$, we compute the the discrete derivative $\Delta_c f(A)$:

$$\Delta_c f(A) = \phi(\mathbf{w}(A) + w[c]) - \phi(\mathbf{w}(A)) \tag{4.4.2}$$

Thus if $A \subseteq B \subseteq E$ and $c \in E \setminus B$, then $\mathbf{w}(A) \leq \mathbf{w}(B)$, so by Eqs. 4.4.1 and 4.4.2, $\Delta_c f(A) \geq \Delta_c f(B)$, and hence f is submodular.

4.4.2 Set Cover Functions as Threshold Potentials

Suppose we are given a finite collection of finite sets $\mathscr{C} = \{C_e\}_{e \in E} \subseteq 2^J$, where E is an index set for the collection, and J is some universal set where all sets in the collection live. Then the collection $\mathscr{C}$ defines a *set cover function*—the cardinality of the union of a subcollection. We represent subcollections by sets of indices $A \in 2^E$.

$$2^E \ni A \mapsto f(A) := \left| \cup_{e \in A} C_e \right|$$

For every $j \in J$, we define the vectors $\mathbf{w}_j \in \{0,1\}^{|E|}$ as the indicator vector of the subcollection of sets that contain j. That is, $w_j[e] = 1$ if $j \in C_e$ and $w_j[e] = 0$ if $j \notin C_e$. We claim:

$$f(A) = \sum_{j \in J} \min(1, \mathbf{w}_j(A))$$

Each term in the sum equals 1 if $j \in \cup_{e \in A} C_e$ and 0 otherwise. Thus summing over all j must give the cardinality of $\cup_{e \in A} C_e$, which is exactly the set cover function.

Hence we conclude all set cover functions can be represented as a sum of threshold potentials with boolean-valued weight vectors and unit thresholds. When $n \geq 3$, this is strictly less general than potentials with boolean weights and nonunit thresholds. For example, $f(A) = \min(2, |A|) = \Psi(2, \mathbb{1}, A)$ cannot be represented as a nonnegative combination of set cover functions.

4.4.3 Reformulation of a Class of Functions

Another example of decomposable functions are the problems under consideration in [II07], which are of the following form:

$$f(A) = \mathbf{u}(A)\phi(\mathbf{v}(A))$$

Where $\mathbf{u}, \mathbf{v}$ are nonnegative weight vectors and ϕ is a nonincreasing concave function. Suppose we can construct a vector $\mathbf{w}_e$ for each $e \in E$ and a concave function ϕ_0 such that the following holds for all $A \in 2^E$:

$$\phi_0(\mathbf{w}_e(A)) = \begin{cases} \phi(\mathbf{v}(A)) - \phi(0) & \text{if } e \in A \\ 0 & \text{if } e \in E \setminus A \end{cases} \tag{4.4.3}$$

Then we claim the following is an equivalent formulation for f in decomposable form:

$$g(A) = \langle \mathbb{1}_A, \phi(0)\mathbf{u} \rangle + \sum_{e \in E} u[e]\phi_0(\mathbf{w}_e(A)) \tag{4.4.4}$$

Indeed, plugging Equation 4.4.3 to the above gives:

$$g(A) = \langle \mathbb{1}_A, \phi(0)\mathbf{u} \rangle + \sum_{e \in A} u[e](\phi(\mathbf{v}(A)) - \phi(0)) = f(A)$$

To satisfy Equation 4.4.3 we define ϕ_0 as follows:

$$\phi_0(t) = \begin{cases} 0 & \text{if } t \leq \mathbf{v}(E) \\ \phi(t - \mathbf{v}(E)) - \phi(0) & \text{if } t > \mathbf{v}(E) \end{cases}$$

And let $\mathbf{w}_e = \mathbf{v} + \mathbf{v}(E)\,\mathbb{1}_e$. It is straightforward to check that these definitions satisfy Equation 4.4.3. Note ϕ_0 is concave because ϕ is nonincreasing concave. Incidentally, the decomposition in Equation 4.4.4 proves that f is submodular.

4.4.4 Strict Generality of Threshold Potentials

Any concave cardinality function can be decomposed into the sum of several threshold potentials with Boolean weight vectors. Equivalently, it is representable as a nonnegative combination of set cover functions.

$$\begin{aligned}\phi(|A\cap B|) = \phi(0) + (\phi(|B|) - \phi(|B|-1))\langle \mathbb{1}_A, \mathbb{1}_B\rangle \\ + \sum_{k=1}^{|B|-1} (2\phi(k) - \phi(k-1) - \phi(k+1)) \min(k, \langle \mathbb{1}_A, \mathbb{1}_B\rangle)\end{aligned}$$

Since ϕ is concave, the coefficients $(2\phi(k) - \phi(k-1) - \phi(k+1))$ are positive. So without loss of generality, any sum of concave cardinality functions can be expressed as a sum of a modular function and a positive combination of threshold potentials with Boolean weights.

$$f(A) = p_0 + \mathbf{p}_1(A) + \sum_{\substack{B\in 2^E \\ |B|\geq 2}} \sum_{k=1}^{|B|-1} r(B,k) \min(k, \langle \mathbb{1}_A, \mathbb{1}_B\rangle) \tag{4.4.5}$$

There are $\sum_{k=2}^{n} \binom{n}{k}(k-1) = 1 + 2^{n-1}(n-2)$ coefficients $r(B,k)$ and they all must be nonnegative. If n is small enough, we can evaluate all 2^n values of a set function and use a linear program solver to find a representation of the above form, if one exists. When $n = 4$, this is an LP feasibility problem with 5 unconstrained variables (the offset p_0 and the modular part $\mathbf{p}_1$), 17 nonnegative variables (the coefficients $r(B,k)$), and 16 equality constraints (the values of f). We discovered that the threshold potential $f(A) = \min(\tau, \mathbf{w}(A))$ with $\mathbf{w} = [1/2, 1/2, 1/2, 1]$ and $\tau = 1$ does not have a feasible solution to Equation 4.4.5.

Note that any decomposable function can be represented as a finite sum of threshold potentials. This is because $\mathbf{w}(A)$ can only equal finitely many values, and for any concave function, we can construct a positive combination of threshold potentials that agrees with the concave function on those values. For any one-dimensional function ϕ and any finite set of arguments $z_1 < \ldots < z_m$, define the piecewise linear interpolant $\overline{\phi}$:

$$\overline{\phi}(z) = p_0 + p_1 z + \sum_{i=2}^{m-1} r_i \min(z, z_i) \tag{4.4.6}$$

$$p_0 = \phi(z_1) - z_1 \frac{\phi(z_2) - \phi(z_1)}{z_2 - z_1}$$

$$p_1 = \frac{\phi(z_m) - \phi(z_{m-1})}{z_m - z_{m-1}}$$
$$r_i = \frac{\phi(z_i) - \phi(z_{i-1})}{z_i - z_{i-1}} - \frac{\phi(z_{i+1}) - \phi(z_i)}{z_{i+1} - z_i} \text{ for } i = 2 \ldots m-1 \tag{4.4.7}$$

It is straightforward to check that $\overline{\phi}(z_i) = \phi(z_i)$ for all z_i, which justifies our claim that $\overline{\phi}$ is an interpolant for ϕ. By construction in Equation 4.4.6, $\overline{\phi}(z)$ is a linear function plus a linear combination of threshold functions. The coefficients r_i in Equation 4.4.7 are positive for all choices of z_i if and only if ϕ is concave. Hence to represent a general concave potential as a sum of threshold potentials, we take $\{z_i\}_{i=1}^m = \{\mathbf{w}(A) \mid A \in 2^E\}$, where m is the number of possible values of $\mathbf{w}(A)$. Thus $\overline{\phi}(\mathbf{w}(A)) = \phi(\mathbf{w}(A))$ for all $A \in 2^E$. However, for general weight vectors, m could equal 2^n. This means that to use the formula in Equation 4.4.6, we need exponentially many threshold potentials to represent the function $\phi(\mathbf{w}(A))$ exactly. So even though general concave functions do not induce a strictly more general class of submodular functions, they can provide a *much* more compact representation for functions in this class.

4.5 The SLG Algorithm for Threshold Potentials

We now present our algorithm for efficient minimization of a decomposable submodular function f based on smoothed convex minimization. We first show how we can efficiently smooth the Lovász extension of f. We then apply accelerated gradient descent to the gradient of the smoothed function. Lastly, we demonstrate how we can often obtain a certificate of optimality that allows us to stop early, drastically speeding up the algorithm in practice.

4.5.1 The Smoothed Extension of a Threshold Potential

The key challenge in our algorithm is to efficiently smooth the Lovász extension of f, so that we can resort to algorithms for accelerated convex minimization. We now show how we can efficiently smooth the *threshold potentials* $\Psi(\tau, \mathbf{w}, A) = \min(\tau, \mathbf{w}(A))$ of Section 4.3, which are simple enough to allow efficient smoothing, but rich enough when combined to express a large class of submodular functions. We denote the base polytope of a single threshold potential $\Psi(\tau, \mathbf{w}, \cdot)$ by $B_{\Psi(\tau,\mathbf{w},\cdot)} = \mathscr{B}(\tau, \mathbf{w})$. The Lovász extension of $\Psi(\tau, \mathbf{w}, \cdot)$ is thus:

$$\tilde{\Psi}(\tau, \mathbf{w}, \mathbf{x}) = \max_{\boldsymbol{\lambda} \in \mathscr{B}(\tau,\mathbf{w})} \langle \mathbf{x}, \boldsymbol{\lambda} \rangle \tag{4.5.1}$$
$$\mathscr{B}(\tau, \mathbf{w}) := \{ \boldsymbol{\lambda} \in \mathbb{R}^n \mid \langle \mathbb{1}, \boldsymbol{\lambda} \rangle = \tau,\ \mathbf{0} \le \boldsymbol{\lambda} \le \mathbf{w} \}$$

We define a smoothed Lovász extension by taking the infimal convolution with the quadratic function $\frac{1}{2\mu}\|\mathbf{x}\|^2$, where μ is a scale parameter.

$$\begin{aligned} \tilde{\Psi}^{\mu}(\tau, \mathbf{w}, \mathbf{x}) &:= \tilde{\Psi}(\tau, \mathbf{w}, \mathbf{x}) \,\square\, \frac{1}{2\mu}\|\mathbf{x}\|^2 \\ &= \min_{\mathbf{z} \in \mathbb{R}^n} \tilde{\Psi}(\tau, \mathbf{w}, \mathbf{z} - \mathbf{x}) + \frac{1}{2\mu}\|\mathbf{z}\|^2 \\ &= \max_{\boldsymbol{\lambda} \in \mathscr{B}(\tau,\mathbf{w})} \langle \mathbf{x}, \boldsymbol{\lambda} \rangle - \frac{\mu}{2}\|\boldsymbol{\lambda}\|^2 \end{aligned} \tag{4.5.2}$$

To give a simple justification for this definition, we note that for small μ, this function approximates the nonsmooth function uniformly in $\mathbf{x}$. We can take an optimal solution to Equation 4.5.1 and use it in Equation 4.5.2 to get the following bound:

$$0 \le \tilde{\Psi}(\tau, \mathbf{w}, \mathbf{x}) - \tilde{\Psi}^{\mu}(\tau, \mathbf{w}, \mathbf{x}) \le \mu\tau/2 \text{ for all } \mathbf{x} \in \mathbb{R}^n$$

Here we use the fact that $\|\boldsymbol{\lambda}\|^2 \le \|\boldsymbol{\lambda}\|_1 \|\boldsymbol{\lambda}\|_\infty \le \tau$ for all $\boldsymbol{\lambda} \in \mathscr{B}(\tau, \mathbf{w})$, since we assumed $\|\mathbf{w}\|_\infty = 1$. We conclude that the smoothed extension converges uniformly to the nonsmooth as $\mu \to 0$.

To compute this function, we note Equation 4.5.2 is equivalent to the projection of $\mathbf{x}/\mu$ onto $\mathscr{B}(\mathbf{w}, y)$. Due to the simple structure of the base polytope, this can be solved efficiently. The optimal base $\boldsymbol{\lambda}^*$ is also the gradient of the smoothed extension:

$$\nabla \tilde{\Psi}^{\mu}(\tau, \mathbf{w}, \mathbf{x}) = \arg\max_{\boldsymbol{\lambda} \in \mathscr{B}(\tau,\mathbf{w})} \langle \mathbf{x}, \boldsymbol{\lambda} \rangle - \frac{\mu}{2}\|\boldsymbol{\lambda}\|^2 = \Pi_{\mathscr{B}(\tau,\mathbf{w})}(\mathbf{x}/\mu). \tag{4.5.3}$$

Recall that $\Pi_C(\mathbf{x}) := \arg\min_{\mathbf{x}' \in C} \|\mathbf{x} - \mathbf{x}'\|$ denotes the projection of $\mathbf{x}$ onto the convex set C. To solve for $\boldsymbol{\lambda}^*$, we form the Lagrangian and derive the dual problem:

$$\tilde{\Psi}^{\mu}(\tau, \mathbf{w}, \mathbf{x}) = \min_{s \in \mathbb{R}, \mathbf{y}_1, \mathbf{y}_2 \ge \mathbf{0}} \left(\max_{\boldsymbol{\lambda} \in \mathbb{R}^n} \langle \mathbf{x}, \boldsymbol{\lambda} \rangle - \frac{\mu}{2}\|\boldsymbol{\lambda}\|^2 + \langle \mathbf{y}_1, \boldsymbol{\lambda} \rangle + \langle \mathbf{y}_2, \mathbf{w} - \boldsymbol{\lambda} \rangle + s(\tau - \langle \mathbb{1}, \boldsymbol{\lambda} \rangle) \right)$$

$$= \min_{t\in\mathbb{R},\mathbf{y}_1,\mathbf{y}_2\geq\mathbf{0}} \frac{1}{2\mu}\|\mathbf{x}-s\mathbb{1}+\mathbf{y}_1-\mathbf{y}_2\|^2+\langle\mathbf{y}_2,\mathbf{w}\rangle+s\tau.$$

In the second step the optimal $\boldsymbol{\lambda}=\frac{1}{\mu}(\mathbf{x}-s\mathbb{1}+\mathbf{y}_1-\mathbf{y}_2)$. By strong duality, if the variables $s,\mathbf{y}_1,\mathbf{y}_2$ are optimal for the dual problem, then $\boldsymbol{\lambda}$ must also be optimal. If we fix s, then we can solve for the optimal variables $\mathbf{y}_1^*$ and $\mathbf{y}_2^*$ componentwise. So we have:

$$\begin{aligned}
\mathbf{y}_1^* &= \max(\mathbf{0},-\mathbf{x}+s^*\mathbb{1})\\
\mathbf{y}_2^* &= \max(\mathbf{0},\mathbf{x}-s^*\mathbb{1}-\mu\mathbf{w})\\
\Rightarrow \boldsymbol{\lambda}^* &= \max(\mathbf{0},\min(\mathbf{w},(\mathbf{x}-s^*\mathbb{1})/\mu)).
\end{aligned} \tag{4.5.4}$$

This expresses $\boldsymbol{\lambda}^*$ as a function of the unknown optimal variable s^*. For the simple case of 2-potentials, we can solve for s^* explicitly and get a closed form expression for the gradient of the smoothed Lovász extension:

$$\nabla\tilde{\Psi}^{\mu}(1,\mathbb{1}_{bc},\mathbf{x})=\begin{cases}\mathbb{1}_b & \text{if } x[b]\geq x[c]+\mu\\ \mathbb{1}_c & \text{if } x[c]\geq x[b]+\mu\\ \frac{1}{2}(\mathbb{1}_{bc}+\frac{1}{\mu}(x[b]-x[c])(\mathbb{1}_b-\mathbb{1}_c)) & \text{if } |x[b]-x[c]|<\mu\end{cases}$$

For higher dimensions, it is easier to compute s^* implicitly from the constraint $\langle\mathbb{1},\boldsymbol{\lambda}^*\rangle=\tau$. We define $\rho^{\mu}_{\mathbf{x},\mathbf{w}}(s)$ to be the value of $\langle\mathbb{1},\boldsymbol{\lambda}(s)\rangle$, where $\boldsymbol{\lambda}(s)$ is given by the formula of Equation 4.5.4.

$$\rho^{\mu}_{\mathbf{x},\mathbf{w}}(s):=\sum_{e\in E}\max(0,\min(w[e],(x[e]-s)/\mu))$$

This function is a monotonic continuous piecewise linear function of s, so one approach is to use a simple root-finding algorithm to solve $\rho^{\mu}_{\mathbf{x},\mathbf{w}}(s^*)=\tau$. Additionally, in Section 5.4, we outline two algorithms for solving this problem. First, we derive an explicit formula that requires sorting $2n$ elements to compute. Also, we show how it is possible to find s^* with a linear number of operations (in expectation) via a randomized algorithm.

4.5.2 The SLG Algorithm for Minimizing Sums of Threshold Potentials

Stepping beyond a single threshold potential, we now assume that the submodular function to be minimized can be written as a nonnegative linear combination of threshold potentials and a modular function, i.e.,

$$f(A) = \mathbf{c}(A) + \sum_j r_j \Psi(\tau_j, \mathbf{w}_j, A). \tag{4.5.5}$$

So the smoothed Lovász extension of f, and its gradient (cf. Equation 4.5.3) are given by:

$$\tilde{f}^{\mu}(\mathbf{x}) = \langle \mathbf{x}, \mathbf{c} \rangle + \sum_j r_j \tilde{\Psi}^{\mu}(\tau_j, \mathbf{w}_j, \mathbf{x}), \tag{4.5.6}$$
$$\nabla \tilde{f}^{\mu}(\mathbf{x}) = \mathbf{c} + \sum_j r_j \Pi_{\mathcal{B}(\tau_j, \mathbf{w}_j)}(\mathbf{x}/\mu)$$

We can use the accelerated gradient descent algorithm of [Nes05] to minimize this function. This algorithm requires that the smoothed objective have a Lipschitz continuous gradient. That is, for some constant L, it must hold that $\|\nabla \tilde{f}^{\mu}(\mathbf{x}) - \nabla \tilde{f}^{\mu}(\mathbf{y})\| \le L\|\mathbf{x}-\mathbf{y}\|$, for all $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$. Fortunately, by construction, the smoothed threshold extensions $\tilde{\Psi}^{\mu}(\tau_j, \mathbf{w}_j, \mathbf{x})$ all have $1/\mu$ Lipschitz gradient, a direct consequence of the characterization in Equation 4.5.3. Hence we have a loose upper bound for the Lipschitz constant of $\tilde{f}^{\mu}$: $L \le \frac{D}{\mu}$, where $D = \sum_j r_j$. Furthermore, the smoothed threshold extensions approximate the threshold extensions uniformly: $|\tilde{\Psi}^{\mu}(\tau_j, \mathbf{w}_j, \mathbf{x}) - \tilde{\Psi}(\tau_j, \mathbf{w}_j, \mathbf{x})| \le \frac{\mu\tau_j}{2}$ for all $\mathbf{x}$, so $|\tilde{f}^{\mu}(\mathbf{x}) - \tilde{f}(\mathbf{x})| \le \frac{\mu D_2}{2}$, where $D_2 = \sum_j r_j \tau_j$.

One way to use the smoothed gradient is to specify an accuracy ε, then minimize $\tilde{f}^{\mu}$ for sufficiently small μ to guarantee that the solution will also be an approximate minimizer of $\tilde{f}$. Then we simply apply the accelerated gradient descent algorithm of [Nes05]. See also [BBC11] for a description. Note that the projection onto the unit cube is trivial: $\Pi_{[0,1]^n}(\mathbf{x}) = \max(\mathbf{0}, \min(\mathbb{1}, \mathbf{x}))$. Algorithm 4.1 formalizes our *Smoothed Lovász Gradient (*SLG*)* algorithm. The optimality gap of a smooth convex function at the iterate $\mathbf{y}_t$ can be computed from its gradient:

$$\text{gap}^k = -\min_{\mathbf{x} \in [0,1]^n} \langle \mathbf{x} - \mathbf{y}^k, \nabla \tilde{f}^{\mu}(\mathbf{y}^k) \rangle = \langle \mathbf{y}^k, \nabla \tilde{f}^{\mu}(\mathbf{y}^k) \rangle + \langle \mathbb{1}, \max(\mathbf{0}, -\nabla \tilde{f}^{\mu}(\mathbf{y}^k)) \rangle.$$

In summary, as a consequence of the results of [Nes05], we have the following guarantee about SLG:

Theorem 4.1. SLG *is guaranteed to provide an ε-optimal solution after running for $\mathcal{O}(\frac{D}{\varepsilon})$*

Algorithm 4.1: SLG: Smoothed Lovász Gradient

```
Input: Accuracy $\varepsilon$; decomposable function $f$.
begin
    $\mu = \frac{\varepsilon}{2D_2}$, $L = \frac{D}{\mu}$, $\mathbf{x}^0 = \mathbf{z}^0 = \frac{1}{2}\mathbb{1}$;
    for $k = 1, 2, \ldots$ do
        $\mathbf{g}^k \leftarrow \nabla \tilde{f}^{\mu}(\mathbf{x}^{k-1})/L$;
        $\mathbf{y}^k \leftarrow \Pi_{[0,1]^n}(\mathbf{x}^{k-1} - \mathbf{g}^k)$;
        $\mathbf{z}^k \leftarrow \Pi_{[0,1]^n}\left(\mathbf{z}^0 - \frac{1}{2}\sum_{j=1}^{k} j\,\mathbf{g}^j\right)$;
        if $gap^k \leq \varepsilon/2$ then stop;
        ;
        $\mathbf{x}^k \leftarrow (k\,\mathbf{y}^k + 2\,\mathbf{z}^k)/(k+2)$;
    $\mathbf{x}_\varepsilon \leftarrow \mathbf{y}^k$;
Output: $\varepsilon$-optimal $\mathbf{x}_\varepsilon$ to $\min_{\mathbf{x}\in[0,1]^n} \tilde{f}(\mathbf{x})$
```

iterations.

SLG is only guaranteed to provide an ε-optimal solution to the *continuous* optimization problem. Fortunately, once we have an ε-optimal point for the Lovász extension, we can efficiently round it to set which is ε-optimal for the original submodular function using Algorithm 4.2. As shown in Equation 4.2.3, the extension $\tilde{f}$ is a *convex* combination of the values of the set function, so it cannot be smaller than the smallest of those values.

Algorithm 4.2: Set Generation by Rounding the Continuous Solution

```
Input: Vector $\mathbf{x} \in [0,1]^n$; submodular function $f$.
begin
    By sorting, find any permutation $\pi$ satisfying: $x[\pi_1] \geq \ldots \geq x[\pi_n]$;
    $\varpi^k \leftarrow \{\pi_1, \ldots, \pi_k\}$;
    $K^* \leftarrow \arg\min_{k\in\{0,1,\ldots,n\}} f(\varpi^k)$;
    $\mathscr{C} \leftarrow \{\varpi^k \mid k \in K^*\}$;
Output: Collection of sets $\mathscr{C}$, such that $f(A) \leq \tilde{f}(\mathbf{x})$ for all $A \in \mathscr{C}$
```

4.5.3 Early Stopping based on Discrete Certificates of Optimality

In general, if the minimum of f is not unique, the output of SLG may be in the interior of the unit cube. However, if f admits a unique minimum A^*, then the iterates will tend toward the corner $\mathbf{e}_{A^*}$. Of course, since we are actually interested in solving a discrete problem, it is not necessary to wait for the iterates to converge to the optimal corner. Below, we show that it is possible to use information about the current iterates to check optimality of a set and terminate the algorithm before the continuous problem has converged.

To prove optimality of a candidate set A, we can use a subgradient of $\tilde{f}$ at $\mathbf{e}_A$, which is also a discrete subgradient for f at A. If $\boldsymbol{\lambda} \in \partial f(A)$, and then we can compute an optimality gap:

$$f(A) - f^* \leq \max_{\mathbf{x} \in [0,1]^n} \langle \mathbb{1}_A - \mathbf{x}, \boldsymbol{\lambda} \rangle = \sum_{a \in A} \max(0, \lambda[a]) + \sum_{b \in E \setminus A} \max(0, -\lambda[b]). \tag{4.5.7}$$

In particular, if $\lambda[a] \leq 0$ for $a \in A$ and $\lambda[b] \geq 0$ for $b \in E \setminus A$, then A is optimal. With only knowledge of candidate set A, finding a subgradient $\boldsymbol{\lambda} \in \partial f(A)$ which demonstrates optimality may be extremely difficult, as the subdifferential is a polyhedron with exponentially many extreme points. But our algorithm naturally suggests the subgradient we could use; the gradient of the smoothed extension is also a discrete sugradient—provided that a particular condition holds, as described in the following Lemma.

Lemma 4.2. *Suppose f is a decomposable submodular function with Lovász smoothed extension $\tilde{f}^\mu$ as in Equation 4.5.5 and Equation 4.5.6, respectively. If $\mathbf{x} \in \mathbb{R}^n$ and $A \in 2^E$ satisfy*

$$\min_{a \in A} x[a] \geq \mu + \max_{b \in E \setminus A} x[b], \tag{4.5.8}$$

then $\nabla \tilde{f}^\mu(\mathbf{x}) \in \partial f(A)$.

Proof. By linearity of subdifferentials, it is sufficient to consider the case where f is single threshold potential. Recall the formula for the gradient:

$$\boldsymbol{\lambda} = \nabla \tilde{\Psi}^\mu(\tau, \mathbf{w}, \mathbf{x}) = \max(\mathbf{0}, \min(\mathbf{w}, (\mathbf{x} - s^* \mathbb{1})/\mu)) \quad s^* \text{ chosen so that } \langle \mathbb{1}, \boldsymbol{\lambda} \rangle = \tau$$

There are two possibilities to consider. First, if $\min_{a \in A} x[a] \leq s^* - \mu$, then Equation 4.5.8 implies for all $b \in E \setminus A$ we have $x[b] < s^*$ and thus $\lambda[k] = 0$. This means that $\boldsymbol{\lambda}(E \setminus A) = 0$ and so $\boldsymbol{\lambda}(A) = \boldsymbol{\lambda}(E) = \tau$. So clearly $\boldsymbol{\lambda}$ is a maximing base for the set A. That is, $\boldsymbol{\lambda} \in \arg\max_{\boldsymbol{\lambda}' \in \mathscr{B}(\tau, \mathbf{w})} \langle \mathbb{1}_A, \boldsymbol{\lambda}' \rangle = \partial \tilde{\Psi}(\tau, \mathbf{w}, \mathbb{1}_A) \subseteq \partial \Psi(\tau, \mathbf{w}, A)$.

On the other hand, if $\min_{a \in A} x[a] > s^* - \mu$, then for all $a \in A$ we have $(x[a] - s^*)/\mu \geq 1 \geq w[a]$ and thus $\lambda[a] = w[a]$. Therefore we must have $\boldsymbol{\lambda}(A) = \mathbf{w}(A)$, and since $\boldsymbol{\lambda}'(A) \leq \mathbf{w}(A)$ for any $\boldsymbol{\lambda}' \in \mathscr{B}(\tau, \mathbf{w})$, again it is true that $\boldsymbol{\lambda}$ is a maximizing base for A. □

Lemma 4.2 states that if the components of point $\mathbf{x}$ corresponding to elements of A are all larger than all the other components by at least μ, then the gradient at $\mathbf{x}$ is a subgradient for $\tilde{f}$ at $\mathbb{1}_A$ (which by Equation 4.5.7 allows us to compute an optimality gap). In practice, this separation of components naturally occurs as the iterates move in the direction of the point $\mathbb{1}_A$, long before they ever actually reach the point $\mathbb{1}_A$. But even if the components are not separated, we can easily add a positive multiple of $\mathbb{1}_A$ to separate them and then compute the gradient there to get an optimality gap. In summary, we have the following algorithm to check the optimality of a candidate set:

Algorithm 4.3: Set Optimality Check

```
Input: Set $A$; decomposable function $f$; scale $\mu$; $\mathbf{x} \in \mathbb{R}^n$.
begin
    $\gamma = \mu + \max_{a \in A, b \in E \setminus A} x[a] - x[b]$;
    $\mathbf{g} = \nabla \tilde{f}^{\mu}(\mathbf{x} + \gamma \mathbb{1}_A)$;
    $\delta = \sum_{a \in A} \max(0, g[a]) + \sum_{b \in E \setminus A} \max(0, -g[b])$;
Output: Optimality gap $\delta \geq f(A) - f^*$
```

Of critical importance is how to choose the candidate set A. However, by Equation 4.5.7, for a set to be optimal, we want the components of the gradient $\nabla \tilde{f}^{\mu}(\mathbf{x}+\gamma \mathbb{1}_A)[a]$ to be negative for $a \in A$ and positive for $b \in E \setminus A$. So it is natural to choose $A = \{a \in E \mid \nabla \tilde{f}^{\mu}(\mathbf{x})[a] \leq 0\}$. Thus, if adding $\gamma \mathbb{1}_A$ does not change the signs of the components of the gradient, then, in fact, we have found an optimal set. We have found this stopping criterion to be quite effective in practice, and we use it in all of our experiments.

4.6 Extension to General Concave Potentials

To extend our algorithm to work on general concave functions, we note that an arbitrary smooth concave function can be expressed as an integral of threshold potential functions $\min(x, y)$. Informally, this is because $\frac{d^2}{dy^2} \min(x, y) = -\delta(y - x)$, the Dirac delta. (To see why, note $\min(x, y)$ is a piecewise linear function with a jump of -1 in its derivative at $x = y$.) So, ignoring boundary terms and integrating by parts we should expect that $\int \phi''(y) \min(x, y) dy = \int \phi'(y) \frac{d}{dy} \min(x, y) dy = -\int \phi(y) \delta(y - x) dx = -\phi(x)$. We state this formally in the following Lemma:

Lemma 4.3. *For $\phi \in C^2([0,T])$,*

$$\phi(x) = \phi(0) + \phi'(T)x - \int_0^T \min(x,y)\phi''(y)dy, \quad \forall x \in [0,T]$$

Proof. This is a straightforward application of integration by parts:

$$\begin{aligned}\int_0^T \min(x,y)\phi''(y)dy &= \int_0^x y\phi''(y)dy + \int_x^T x\phi''(y)dy \\ &= (y\phi'(y) - \phi(y))\big|_0^x + x\phi'(y)\big|_x^T \\ &= x\phi'(x) - \phi(x) + \phi(0) + x\phi'(T) - x\phi'(x) \\ &= \phi(0) + x\phi'(T) - \phi(x).\end{aligned}$$

□

This Lemma motivates our definition of the smoothed Lovász extension for a general sum of concave potentials. If f is decomposable as in Equation (4.3.1), we define the smoothed extension as:

$$\tilde{f}^\mu(\mathbf{x}) := \langle \mathbf{c}, \mathbf{x} \rangle + \sum_j \left(\phi_j(0) + \phi'(T_j)\langle \mathbf{x}, \mathbf{w}_j \rangle - \int_0^{T_j} \tilde{\Psi}^\mu(\tau, \mathbf{w}_j, \mathbf{x})\phi_j''(\tau)d\tau \right). \tag{4.6.1}$$

Here $T_j := \mathbf{w}_j(E)$. By Lemma 4.3, as $\mu \to \infty$, we have $\tilde{f}^\mu \to \tilde{f}$ pointwise since $\tilde{\Psi}^\mu \to \tilde{\Psi}$. We can apply SLG to $\tilde{f}^\mu$ essentially unchanged; the conditions for optimality still hold, and so on. Conceptually, we just use a different smoothed gradient, but calculating it is more involved. We need to compute the integrals of the form $\int_0^{T_j} \nabla\tilde{\Psi}^\mu(\tau, \mathbf{w}_j, \mathbf{x})\phi_j''(\tau)d\tau$. Since the components of the gradient $\nabla\tilde{\Psi}^\mu(\tau, \mathbf{w}_j, \mathbf{x})$ are each piecewise linear functions with respect to τ, we can evaluate the integral by parts. The resulting formula only evaluates ϕ, but not its derivatives.

4.6.1 Formula Derivation

Let $f(A) = \phi(\mathbf{w}(A))$ be a general concave potential. For ease of notation, in the following let $\mathbf{g}(\tau) = \nabla\tilde{\Psi}^\mu(\tau, \mathbf{w}, \mathbf{x})$ be the gradient of the smoothed extension of a threshold potential. Then by Equation 4.6.1, the gradient of smoothed extention of f is given by:

$$\nabla\tilde{f}^\mu(\mathbf{x}) = \phi'(T)\mathbf{w} - \int_0^T \mathbf{g}(\tau)\phi''(\tau)d\tau \tag{4.6.2}$$

Note that $\mathbf{g}$ is a piecewise linear function of τ. Let $[\tau_i, \tau_{i+1}]$ with $0 = \tau_0 \leq \ldots \leq \tau_N = T = \mathbf{w}(E)$ be the intervals on which $\mathbf{g}$ is linear. Let $\mathbf{g}_i := \mathbf{g}(\tau_i)$. In particular, $\mathbf{g}_0 = \mathbf{0}$ and $\mathbf{g}_N = \mathbf{w}$. Denote the derivative of $\mathbf{g}(\tau)$ with respect to τ (wherever it exists) as $\mathbf{g}'(\tau)$. So within the interval (τ_i, τ_{i+1}) it is:

$$\mathbf{g}'(\tau_i + \epsilon) = \frac{\mathbf{g}_{i+1} - \mathbf{g}_i}{\tau_{i+1} - \tau_i} = \mathbf{g}'(\tau_{i+1} - \epsilon) \tag{4.6.3}$$

So then our smoothed gradient can be evaluated:

$$\begin{aligned}
\nabla \tilde{f}^{\mu}(\mathbf{x}) &= \phi'(\tau_N)\mathbf{w} - \sum_{i=1}^{N} \int_{\tau_{i-1}}^{\tau_i} \mathbf{g}(\tau)\phi''(\tau)d\tau \\
&= \phi'(\tau_N)\mathbf{w} + \sum_{i=1}^{N} \left(\mathbf{g}'(\tau)\phi(\tau) - \mathbf{g}(\tau)\phi'(\tau)\right)\Big|_{\tau_{i-1}}^{\tau_i} \\
&= \phi'(\tau_N)\mathbf{w} + \sum_{i=1}^{N} \mathbf{g}'(\tau_i - \epsilon)(\phi(\tau_i) - \phi(\tau_{i-1})) - \sum_{i=1}^{N} (\mathbf{g}_i\phi'(\tau_i) - \mathbf{g}_{i-1}\phi'(\tau_{i-1})) \\
&= \phi'(\tau_N)\mathbf{w} + \sum_{i=1}^{N} \frac{(\mathbf{g}_i - \mathbf{g}_{i-1})(\phi(\tau_i) - \phi(\tau_{i-1}))}{\tau_i - \tau_{i-1}} - (\mathbf{g}_N\phi'(\tau_N) - \mathbf{g}_0\phi'(\tau_0)) \\
&= \sum_{i=1}^{N} \frac{(\mathbf{g}_i - \mathbf{g}_{i-1})(\phi(\tau_i) - \phi(\tau_{i-1}))}{\tau_i - \tau_{i-1}}
\end{aligned}$$

Interestingly, this formula no longer requires ϕ to be differentiable. To compute the breakpoints for τ, note that τ is itself a piecewise linear function of the parameter s, implicitly defined by the equation: $\langle \mathbb{1}, \max(\mathbf{0}, \min(\mathbf{w}, (\mathbf{x} - s\mathbb{1})/\mu))\rangle = \tau$. The discontinuities of the derivative occur where $x[e] - s$ equals 0 or $\mu w[e]$. Let s_i be the value of s that corresponds to τ_i. For a particular value of s, we say that the component of $\mathbf{x}$ indexed by e is an *active* component if $0 \leq x[e] - s \leq \mu w[e]$. Thus the difference $\mathbf{g}_i - \mathbf{g}_{i-1}$ is nonzero only in the active components, and there it equals $(s_i - s_{i-1})/\mu$. Let A_i be the set of active components for $\tau \in [\tau_{i-1}, \tau_i]$. Then, in that interval we have $\frac{d\tau}{ds} = |A_i|/\mu$ and we can simplify a part of the above sum:

$$\frac{\mathbf{g}_i - \mathbf{g}_{i-1}}{\tau_i - \tau_{i-1}} = \frac{\mathbb{1}_{A_i}(s_i - s_{i-1})/\mu}{(s_i - s_{i-1})|A_i|/\mu} = \frac{\mathbb{1}_{A_i}}{|A_i|}. \tag{4.6.4}$$

In summary, the formulas reduce to:

$$P = \{(-x[e], e), (\mu w[e] - x[e], e)\}_{e \in E} = \{(s_i, a_i)\}, \text{ where } s_i \leq s_{i+1} \tag{4.6.5}$$

$$A_i = a_1 \ominus a_2 \ldots \ominus a_i \tag{4.6.6}$$

$$\tau_j = \sum_{i=1}^{j} |A_i|(s_{i+1} - s_i)/\mu \tag{4.6.7}$$

$$\nabla \tilde{f}^{\mu}(\mathbf{x}) = \sum_{i=1}^{2n-1} \mathbb{1}_{A_i} \frac{\phi(\tau_i) - \phi(\tau_{i-1})}{|A_i|} \tag{4.6.8}$$

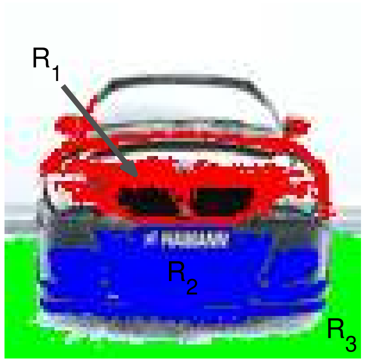
A naïve implementation would compute and add each vector in turn, and thus require $O(n^2)$ operations in the worst case. This is because there is no guarantee that the sets A_i are small. In fact, it may be that entire ground set is active and thus $\mathbb{1}_{A_i}$ fully dense. Despite this, it is possible to compute the sum with only $O(n)$ operations (not counting the cost of sorting). To see how we can do this, first define $z_j := \sum_{i=1}^{j} \frac{\phi(\tau_i)-\phi(\tau_{i-1})}{|A_i|}$. Then any component of $\mathbf{g}$ is a difference $z_j - z_{j'}$ for some indices j, j'. Namely, j is the step at which the component activates, and j' is the step at which it deactivates. The entire procedure is listed in Algorithm 4.4.

Algorithm 4.4: Gradient for General Concave Functions

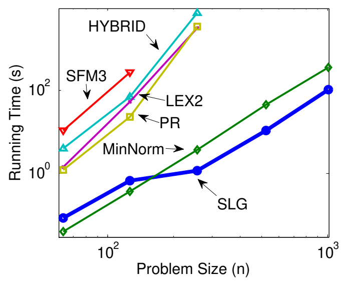
```
Input: Domain vector $\mathbf{x}$, Weight vector $\mathbf{w}$, Concave function $\phi$, Smoothing scale $\mu$.
begin
    $P \leftarrow \cup_{e \in E}\{(-x[e], e), (\mu w[e] - x[e], e)\}$;
    Sort $P = \{(s_i, a_i)\}$, where $s_i \leq s_{i+1}$;
    $\tau_0 \leftarrow 0$; $z \leftarrow 0$; $A \leftarrow \emptyset$;
    for $i = 1 \ldots 2n-1$ do
        if $a_i \notin A$ then
            $g[a_i] \leftarrow -z$;
            $A \leftarrow A + a_i$;
        else
            $g[a_i] \leftarrow g[a_i] + z$;
            $A \leftarrow A - a_i$;
        $\tau_i \leftarrow \tau_{i-1} + |A|(s_{i+1} - s_i)/\mu$;
        if $A \neq \emptyset$ then
            $z \leftarrow z + \big(\phi(\tau_i) - \phi(\tau_{i-1})\big)/|A|$;
    $g[a_{2n}] \leftarrow g[a_{2n}] + z$;
Output: $\mathbf{g}$ = Gradient for $\mu$–smoothed Lovász extension of $\phi(\mathbf{w}(A))$ at $\mathbf{x}$
```

4.7 Experiments

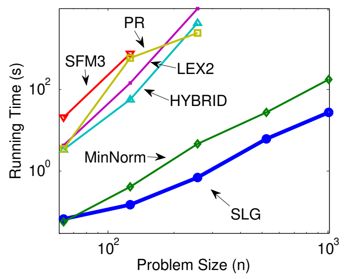
Synthetic Data. We reproduce the experimental setup of [FI11] designed to compare submodular minimization algorithms. Our goal is to find the minimum cut of a randomly



(a) *Example Regions for Potentials*



(b) *Results for genrmf-long*

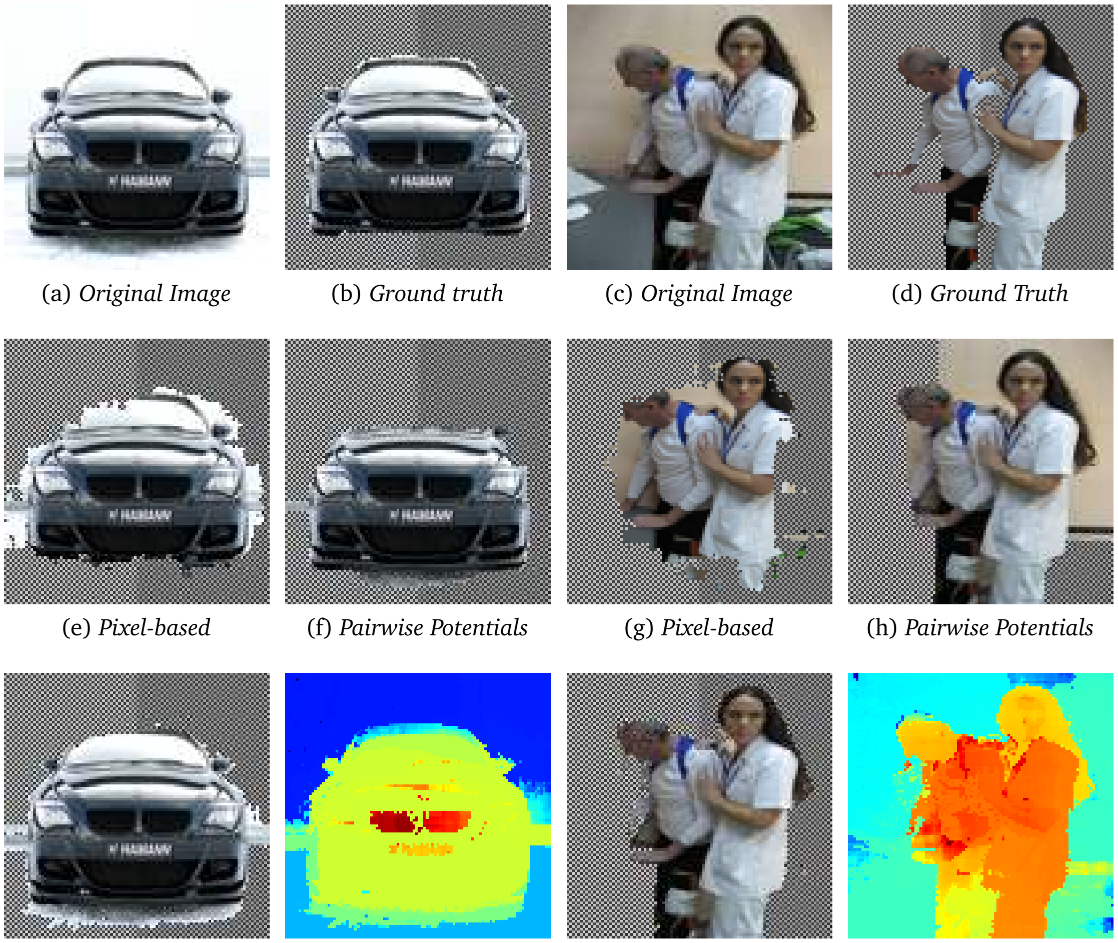


(c) *Results genrmf-wide*

Figure 4.1: (a) Example regions used for our higher-order potential functions (b-c) Comparision of running times of submodular minimization algorithms on synthetic problems from DIMACS [JM93].

generated graph (which requires submodular minimization of a sum of 2-potentials) with the graph generated by the specifications in [JM93]. We compare against the state of the art combinatorial algorithms (LEX2, HYBRID, SFM3, PR [FI03]) that are guaranteed to find the exact solution in polynomial time, as well as the Minimum Norm algorithm of [FI11], a practical alternative with unknown running time. Figures 4.1b and 4.1c compare the running time of SLG against the running times reported in [FI11]. In some cases, SLG was 6 times faster than the MinNorm algorithm. However, the comparison to the MinNorm algorithm is inconclusive in this experiment, because while we used a faster machine, we also used a simple MATLAB implementation. What is clear is that SLG scales at least as well as MinNorm on these problems, and is practical for problem sizes that the combinatorial algorithms cannot handle.

Image Segmentation Experiments. We also tested our algorithm on the joint image segmentation-and-classification task introduced in Section 4.3. We used an implementation of TextonBoost [SWRC09], then trained on and tested subsampled images from [EVGW$^+$]. As seen in Figures 4.2e and 4.2g, using only the per-pixel score from our TextonBoost implementation gets the general area of the object, but does not do a good job of identifying the shape of a classified object. Compare to the ground truth in Figures 4.2b and 4.2d. We then perform MAP inference in a Markov Random Field with 2-potentials (as done in [SWRC09]). While this regularization, as shown in Figures 4.2f and 4.2h, leads to improved performance, it still performs poorly on classifying the boundary.



(a) *Original Image* (b) *Ground truth* (c) *Original Image* (d) *Ground Truth*

(e) *Pixel-based* (f) *Pairwise Potentials* (g) *Pixel-based* (h) *Pairwise Potentials*

(i) *Concave Potentials* (j) *Continuous* (k) *Concave Potentials* (l) *Continuous*

Figure 4.2: Segmentation Experimental Results

Finally, we used SLG to regularize with higher order potentials. To generate regions for our potentials, we randomly picked seed pixels and grew the regions based on HSV channels of the image. We picked our seed pixels with a preference for pixels which were included in the least number of previously generated regions. Figure 4.1a shows what the regions typically looked like. For our experiments, we used 90 total regions. We used SLG to minimize $f(A) = \mathbf{c}(A) + \sum_j |A \cap D_j||D_j \setminus A|$, where $\mathbf{c}$ was the output from TextonBoost, scaled appropriately. Figures 4.2i and 4.2k show the classification output. The continuous variables $\mathbf{x}$ at the end of each run are shown in Figures 4.2j and 4.2l; while it has no formal meaning, in general one can interpret a very high or low value of $x[e]$ to correspond to high confidence in the classification of the pixel e. To generate the result shown in Figure 4.2k, a problem with 10^4 variables and 90 concave potentials, our MATLAB/mex implementation of SLG took 71.4 seconds. In comparison, the MinNorm implementation of the SFO toolbox [Kra10] gave the same result, but took 6900 seconds. Similar problems on an image of twice the resolution (4×10^4 variables) were tested using SLG, resulting in a runtimes of roughly 1600 seconds.

4.8 Conclusion

We have developed a novel method for efficiently minimizing a large class of submodular functions of practical importance. We do so by decomposing the function into a sum of threshold potentials, whose Lovász extensions are convenient for using modern smoothing techniques of convex optimization. This allows us to solve submodular minimization problems with thousands of variables that cannot be expressed using only pairwise potentials. Thus we have achieved a middle ground between graph-cut-based algorithms, which are extremely fast but only able to handle very specific types of submodular minimization problems, and combinatorial algorithms, which assume nothing but submodularity but are impractical for large-scale problems.

Chapter 5

Distributed Submodular Minimization

5.1 Introduction

Without any doubt, there have been tremendous gains in the efficiency of convex optimization algorithms over the past century. The fact that one can now routinely solve nonlinear constrained programs with thousands of variables is not due to advances in hardware alone. However, there are still domains where the problem is so large that a straightforward application of optimization algorithms is not practical. This is often due to an abundance of data; the entire convex optimization problem cannot even be loaded in memory on a single computer, let be alone solved in a reasonable amount of time. For such problems, the only option is to develop algorithms that can take advantage of parallel computation. This is a very active field of research in convex optimization. Given the strong connections between convex and submodular minimization established in Chapter 3, a natural idea, then, is to use techniques of distributed convex optimization for distributed submodular minimization.

First, in Section 5.2, we derive a general framework for submodular minimization using minimization of the Lovász extension with barrier functions. This is not specific to distributed computation, but it suggests a general technique for developing submodular minimization algorithms. Then in Section 5.3, we use this framework by applying accelerated first-order minimization methods to the resulting convex problems to develop submodular minimization algorithms that can be run in parallel. Finally, in Section 5.5, we give some basic empirical results of these algorithms on synthetic test problems.

5.2 Submodular Minimization with General Barrier Functions

In this section, we dissect the different ways one can formulate submodular minimization as a convex optimization program. Let f be a submodular set function over the set E with $|E| = n$. We assume f is normalized, meaning that $f(\emptyset) = 0$. We wish to find a minimizer of f—any subset of E that achieves the minimal value of f:

$$f^* := \min_{A \in 2^E} f(A) \tag{5.2.1}$$

Recall that a submodular function f defines a convex function over $\mathbb{R}^n$ called the Lovász extension, which we denote $\tilde{f}$. It is defined as the support function of the base polytope B_f. That is: $\tilde{f}(\mathbf{x}) = \sigma_{B_f}(\mathbf{x}) = \max_{\boldsymbol{\lambda} \in B_f} \langle \mathbf{x}, \boldsymbol{\lambda} \rangle$. The base polytope is defined by:

$$B_f = \{ \boldsymbol{\lambda} \in \mathbb{R}^n \mid \langle \mathbb{1}, \boldsymbol{\lambda} \rangle = f(E), \text{ and } \langle \mathbb{1}_A, \boldsymbol{\lambda} \rangle \leq f(A) \text{ for all } A \in 2^E \}.$$

Furthermore, minimizing the Lovász extension over the unit cube $[0,1]^n$ is equivalent to minimizing the submodular function in the following sense: given some $\mathbf{x}^* \in [0,1]^n$ that minimizes $\tilde{f}$ and any threshold $\alpha \in (0,1)$, the set of components of $\mathbf{x}^*$ greater than or equal to that threshold must be a minimizer of f. Note that by using an indicator function, we can express the constrained program of minimizing over the unit cube as an unconstrained convex program: $\min_{\mathbf{x}} \tilde{f}(\mathbf{x}) + \delta_{[0,1]^n}(\mathbf{x})$. In the unconstrained program, the unit cube indicator $\delta_{[0,1]^n}$ acts as a barrier to ensure that a minimal $\mathbf{x}$ exists. Without it, the Lovász extension has no finite minimizer in general. That is, unless $\mathbf{0} \in B_f$ (or equivalently $f(E) = f^* = 0$), we have $\inf_{\mathbf{x} \in \mathbb{R}^n} \tilde{f}(\mathbf{x}) = -\infty$. Hence we can consider minimizing $\tilde{f}$ over the unit cube to be a particular case of a more general technique of minimizing $\tilde{f}$ plus a convex barrier function ϕ:

$$\min_{\mathbf{x}} \tilde{f}(\mathbf{x}) + \phi(\mathbf{x}) \tag{5.2.2}$$

We refer to Equation 5.2.2 as the primal program. Its corresponding dual program is:

$$\max_{\boldsymbol{\lambda} \in B_f} -\phi^*(-\boldsymbol{\lambda}) \tag{5.2.3}$$

It is not immediately obvious that if we let ϕ be anything other than the indicator of the unit cube that solving Equation 5.2.2 will lead to a solution of Equation 5.2.1. Indeed, we do need a few conditions beyond convexity to ensure that this is the case. The first three of the of following Properties will ensure that a minimizer of f can be computed by thresholding a solution to the primal problem. The fourth Property is only necessary to ensure that solutions of the dual problem will also give minimizers of f via thresholding.

Properties 5.1. Barrier Function ϕ with Threshold α

5.1.a The function ϕ is separable; it can be expressed as a sum of functions each depending on a single coordinate. This means its conjugate ϕ^* is separable as well.

5.1.b All vectors in $\mathbb{R}^n$ are subgradients of ϕ somewhere, which means ϕ^* has full domain.

5.1.c At some multiple of the ones vector ($\alpha\mathbb{1}$), the only subgradient of ϕ is the zero vector. This means that $\alpha\mathbb{1}$ is a strict subgradient of ϕ^* at the origin.

5.1.d [Optional] The function ϕ is uniquely minimized at $\alpha\mathbb{1}$. This means that $\alpha\mathbb{1}$ is the unique subgradient of ϕ^* at the origin.

In summary:

$$\phi(\mathbf{x}) = \sum_{e\in E} \phi_e(x[e]) \quad \Leftrightarrow \quad \phi^*(\boldsymbol{\lambda}) = \sum_{e\in E} \phi_e^*(\lambda[e]) \tag{5.2.4a}$$

$$\bigcup_{\mathbf{x}\in\operatorname{dom}(\phi)} \partial\phi(\mathbf{x}) = \mathbb{R}^n, \quad \Leftrightarrow \quad \operatorname{dom}(\phi^*) = \mathbb{R}^n \tag{5.2.4b}$$

$$\partial\phi(\alpha\mathbb{1}) = \{\mathbf{0}\} \quad \Leftrightarrow \quad \phi^*(\boldsymbol{\lambda}) > \langle \alpha\mathbb{1}, \boldsymbol{\lambda}\rangle + \phi^*(\mathbf{0}) \text{ for all } \boldsymbol{\lambda} \neq \mathbf{0} \tag{5.2.4c}$$

$$\phi(\mathbf{x}) > \phi(\alpha\mathbb{1}) \text{ for all } \mathbf{x} \neq \alpha\mathbb{1} \quad \Leftrightarrow \quad \partial\phi^*(\mathbf{0}) = \{\alpha\mathbb{1}\} \tag{5.2.4d}$$

For example, $\phi(\mathbf{x}) = \delta_{[0,1]^n}(\mathbf{x})$ satisfies Properties 5.1.a through 5.1.c with any $\alpha \in (0,1)$, but violates Property 5.1.d. The functions $\phi(\mathbf{x}) = \frac{1}{2}\|\mathbf{x}\|^2, -\sum_{e\in E}\log(1-x[e]^2)$ satisfy all the Properties with $\alpha = 0$. As counterexamples, note that $\phi(\mathbf{x}) = \delta_{\{\mathbf{0}\}}(\mathbf{x})$ violates 5.1.c, whereas $\phi(\mathbf{x}) \equiv 0$ violates b and d, and $\phi(\mathbf{x}) = \|\mathbf{x}\|_1$ violates b and c.

The purpose of Property 5.1.c is to ensure that the order of the components of the optimal dual vector are nearly the same as the ordering of the components of the optimal primal vector. Specifically, we will make use of the following Lemma, whose proof is obvious:

Lemma 5.1. *If g is a one-dimensional convex function and $\partial g(\alpha) = \{\beta\}$ and $y \in \partial g(x)$, then $y < \beta$ only if $x < \alpha$, and $y > \beta$ only if $x > \alpha$. Conversely, we have $x < \alpha$ only if $y \leq \beta$, and $x > \alpha$ only if $y \geq \beta$.*

We will first state what we can conclude about the solutions to Equation 5.2.2 and Equation 5.2.3 under the weaker assumption that all but the last of Properties 5.1 hold. We state these results separately but prove them together, because the convex programs are dual to each other.

Proposition 5.2 (Primal form). *Suppose $\tilde{f}$ is the Lovász extension of the submodular function f, and $\phi \in \overline{\mathrm{Conv}}$ satisfies Properties 5.1.a through 5.1.c with threshold α. Let $\mathbf{x}^*$ be a minimizer of Equation 5.2.2. Define the set X_α^{++} (resp. X_α^{+}) to be the set of components of $\mathbf{x}^*$ greater than (resp. greater than or equal to) the threshold α.*

$$\begin{aligned} X_\alpha^{++} &:= \{\, e \in E \mid x^*[e] > \alpha \,\} \\ X_\alpha^{+} &:= \{\, e \in E \mid x^*[e] \geq \alpha \,\} \end{aligned} \tag{5.2.5}$$

Then X_α^{+} and X_α^{++} are minimizers of f. That is, $f(X_\alpha^{+}) = f(X_\alpha^{++}) = f^*$.

Proposition 5.3 (Dual Form). *Suppose $\phi \in \overline{\mathrm{Conv}}$ satisfies Properties 5.1.a through 5.1.c. Let $\boldsymbol{\lambda}^*$ be a maximizer of Equation 5.2.3, wherein B_f is the base polytope of the submodular function f. Define the set Λ_0^{--} (resp. Λ_0^{-}) to be the set of components of $\boldsymbol{\lambda}^*$ which are negative (resp. nonpositive).*

$$\begin{aligned} \Lambda_0^{--} &:= \{\, e \in E \mid \lambda^*[e] < 0 \,\} \\ \Lambda_0^{-} &:= \{\, e \in E \mid \lambda^*[e] \leq 0 \,\} \end{aligned} \tag{5.2.6}$$

Then Λ_0^{-} and Λ_0^{--} are respectively upper and lower bounds for minimizers of f. That is, $f(A) = f^*$, only if $\Lambda_0^{--} \subseteq A \subseteq \Lambda_0^{-}$.

Proof. Since $\operatorname{dom} \tilde{f}^*$ is a polyhedron and $\operatorname{dom} \phi^* = \mathbb{R}^n$, we can apply Fenchel's Duality Theorem to the functions $\tilde{f}$ and ϕ, which gives us:

$$\min_{\mathbf{x} \in \mathbb{R}^n} \tilde{f}(\mathbf{x}) + \phi(\mathbf{x}) = \max_{\boldsymbol{\lambda} \in \mathbb{R}^n} -\tilde{f}^*(\boldsymbol{\lambda}) - \phi^*(-\boldsymbol{\lambda})$$

Since $\tilde{f} = \sigma_{B_f}$, and $\tilde{f}^* = \delta_{B_f}$, the right hand side is equivalent to Equation 5.2.3. By Equation 2.1.5, the optimal solutions $(\mathbf{x}^*, \boldsymbol{\lambda}^*)$ of these programs are characterized by:

$$\boldsymbol{\lambda}^* \in \partial \tilde{f}(\mathbf{x}^*) \cap -\partial \phi(\mathbf{x}^*), \qquad \mathbf{x}^* \in \partial \tilde{f}^*(\boldsymbol{\lambda}^*) \cap \partial \phi^*(-\boldsymbol{\lambda}^*).$$

This implies $-\lambda^*[e] \in \partial \phi_e(x^*[e])$, where ϕ_e is the one-dimensional convex function from the decomposition $\phi(\mathbf{x}) = \sum_{e\in E} \phi_e(x[e])$. Thus by Lemma 5.1 and Property 5.1.c we have:

$$\begin{array}{ccccccc} e \in \Lambda_0^{--} & \Leftrightarrow & \lambda^*[e] < 0 & \Rightarrow & x^*[e] > \alpha & \Leftrightarrow & e \in X_\alpha^{++} \\ e \notin \Lambda_0^{-} & \Leftrightarrow & \lambda^*[e] > 0 & \Rightarrow & x^*[e] < \alpha & \Leftrightarrow & e \notin X_\alpha^{+} \end{array}$$

This means that $\Lambda_0^{--} \subseteq X_\alpha^{++} \subseteq X_\alpha^{+} \subseteq \Lambda_0^{-}$, hence:

$$\lambda^*[e] \geq 0 \text{ for all } e \in X_\alpha^+, \qquad \lambda^*[e] \leq 0 \text{ for all } e \in E \setminus X_\alpha^{++}.$$

Also note by Equation 3.3.9, we have $\boldsymbol{\lambda}^* \in \partial\tilde{f}(\mathbf{x}^*) \subseteq \partial f(X_\alpha^+)$. So the set X_α^+ and base $\boldsymbol{\lambda}^*$ satisfy the premise of Theorem 3.7, and so X_α^+ must be a minimizer of f. Likewise, since $X_\alpha^{++} = X_\beta^+$ for some $\beta > \alpha$, we can use Equation 3.3.9 to conclude $\boldsymbol{\lambda}^* \in \partial f(X_\alpha^{++})$, and so by the same argument X_α^{++} is a minimizer of f.

To prove Proposition 5.3, we use the same facts we established in the proof of Proposition 5.2. Namely, since $f(X_\alpha^+) = f^*$, $\boldsymbol{\lambda}^* \in \partial f(X_\alpha^+)$, and $\Lambda_0^{--} \subseteq X_\alpha^+ \subseteq \Lambda_0^-$, we can bound $f(A)$ for any $A \in 2^E$:

$$\begin{aligned} f(A) - f^* \geq \langle \mathbb{1}_A - \mathbb{1}_{X_\alpha^+}, \boldsymbol{\lambda}^* \rangle &= \sum_{e\in A\setminus X_\alpha^+} \lambda^*[e] - \sum_{e\in X_\alpha^+\setminus A} \lambda^*[e] \\ &= \sum_{e\in A\setminus \Lambda_0^-} |\lambda^*[e]| + \sum_{e\in \Lambda_0^{--}\setminus A} |\lambda^*[e]| \end{aligned}$$

Since $|\lambda^*[e]| > 0$ for $e \in E \setminus (\Lambda_0^- \setminus \Lambda_0^{--})$, if either $A \setminus \Lambda_0^-$ or $\Lambda_0^{--} \setminus A$ is nonempty we have $f(A) > f^*$. Therefore, any minimizer of f must be sandwiched between Λ_0^{--} and Λ_0^- as claimed. □

Note that if no component of $\boldsymbol{\lambda}^*$ equals zero, then $\Lambda_0^{--} = \Lambda_0^-$ is the unique minimizer of f. In general though, a solution to the dual problem might only give nonstrict bounds for minimizers of f. For example, suppose $E = \{1,2\}$ and $f(A) = \min(1, |A|)$. Then $A^* = \emptyset$ is the unique minimizer of f and $B_f = \{(\theta, 1-\theta) \mid \theta \in [0,1]\}$. Suppose we use the barrier function $\phi = \delta_{[0,1]^2}$. Then $\phi^*(-\boldsymbol{\lambda}) = \sigma_{[0,1]^2}(-\boldsymbol{\lambda}) = 0$ for all $\boldsymbol{\lambda} \in B_f$. If we take $\boldsymbol{\lambda}^* = (1,0)$, then $\Lambda_0^- = \{2\}$, which is not a minimizer of f. (This does not contradict the Theorem since $\Lambda_0^{--} = \emptyset$, hence $\Lambda_0^{--} \subseteq A^* \subseteq \Lambda_0^-$.) Likewise, it is also possible to construct an example where Λ_0^{--} is not a minimizer. Consider an example identical to the previous one except that $f(A) = \min(1, |A|) - |A|$. In this case, $A^* = \{1,2\}$ is the unique minimizer of f, and $\boldsymbol{\lambda}^* = (-1,0)$ is maximal for Equation 5.2.3. But then $\Lambda_0^{--} = \{1\}$ is not a minimizer of f. Finally, note that if we take the direct product of these two examples, then we get a case where $\Lambda_0^{--} \subsetneq A^* \subsetneq \Lambda_0^{-}$.

These counterexamples are due to the fact that if ϕ does not have a unique minimizer, we do not have a strong enough condition on the components $x^*[e]$ for which $\lambda^*[e] = 0$. However, when ϕ is is uniquely minimized at $\alpha\mathbb{1}$, then the primal and dual problems give identical results.

Theorem 5.4. *Suppose $\phi \in \overline{\mathrm{Conv}}$ satisfies Properties 5.1. Let $\mathbf{x}^*$ be a minimizer of Equation 5.2.2 and use it to define $X_\alpha^{++}, X_\alpha^{+}$ as in Equation 5.2.5. Let $\boldsymbol{\lambda}^*$ be a maximizer of Equation 5.2.3 and use it to define $\Lambda_0^{--}, \Lambda_0^{-}$ as in in Equation 5.2.6. Then $X_\alpha^{++} = \Lambda_0^{--}$ is the unique minimal minimizer of f, and $X_\alpha^{+} = \Lambda_0^{-}$ is the unique maximal minimizer of f.*

Proof. The hypothesis of the Theorem is identical to the Propositions, with the additional assumption that Property 5.1.d holds. This extra condition implies $0 \notin \partial\phi_e(x)$ for $x \neq \alpha$. Hence:

$$-\lambda \in \partial\phi_e(x) \Rightarrow \begin{cases} x < \alpha \iff \lambda > 0 \\ x = \alpha \iff \lambda = 0 \\ x > \alpha \iff \lambda < 0 \end{cases}$$

Thus, since $-\boldsymbol{\lambda}^* \in \partial\phi(\mathbf{x}^*)$, we must have $X_\alpha^{++} = \Lambda_0^{--}$ and $X_\alpha^{+} = \Lambda_0^{-}$. The theorem then follows immediately from the conclusions of Proposition 5.2 and Proposition 5.3. □

Discussion The implication of this result is that we have a great deal of flexibility in how we can solve Equation 5.2.1 through convex optimization. Any barrier function which satisfies the conditions of the theorem can be used.

Properties 5.1 are, in some sense, necessary for the conclusions of the Theorem to hold. Clearly B_f must be contained in $\operatorname{dom}\phi^*$, and Property 5.1.b is needed to ensure this is true for all submodular functions f. Because Property 5.1.d is not necessary to find a minimizer of f with the primal problem, one might suspect by symmetry that Properties 5.1.a, b and d would be sufficient to find a minimizer of f with the dual problem. This is not the case, however. For example, if we tried to use $\phi(\mathbf{x}) = \varepsilon\|\mathbf{x}\|_1 + \frac{1}{2}\|\mathbf{x}\|^2$, and $B_f \subseteq [-\varepsilon, \varepsilon]^n$, then we would have $\mathbf{x}^* = 0$, and $\boldsymbol{\lambda}^*$ an arbitrary point in B_f, so solving this convex program would tell us nothing useful about the the submodular minimization problem.

Note that we define the Lovász extension to be the support function of the submodular polytope B_f rather than the submodular polyhedron P_f. Using the base polytope is strictly more general in the following sense: the Theorem still holds with P_f used in place of B_f, provided that the threshold $\alpha > 0$. To see this, first note that $\sigma_{P_f} = \sigma_{B_f} + \delta_{\mathbb{R}^n_+}$. Hence minimizing $\sigma_{P_f} + \phi_1$ is equivalent to minimizing $\sigma_{B_f} + \phi_2$, where $\phi_2 = \phi_1 + \delta_{\mathbb{R}^n_+}$, and $\phi_2^* = \phi_1^* \,\square\, \delta_{\mathbb{R}^n_-}$. Then ϕ_2 satisfies Properties 5.1 if and only if ϕ_1 does with $\alpha > 0$.

5.3 Consensus Algorithms for Submodular Minimization

5.3.1 Notation

Using convex analysis, we can develop consensus algorithms for submodular minimization. In particlar, suppose our submodular functions can be reprsented as a sum of a modular function plus M submodular functions, each of which depends on a subset of the ground set.

$$\min_A \; \mathbf{c}(A) + \sum_{i=1}^{M} f_i(A \cap E_i)$$

The assumption that each subfunction f_i depends only on a subset of the variables confers no loss of generality, since we can just take each E_i to be the full ground set E if necessary. However, it can lead to a substantially more efficient algorthm (in time and memory) in the case where the average coverage $\frac{1}{M}\sum_{i=1}^{M} |E_i|/|E|$ is small.

By Theorem 5.4, we can minimize f by computing the proximal operator of the Lovász extension $\tilde{f}$:

$$\min_{\mathbf{x}\in\mathbb{R}^n} \; \frac{1}{2}\,||\mathbf{x}||^2 + \langle \mathbf{x}, \mathbf{c}\rangle + \sum_i \tilde{f}_i(\mathbf{P}_i\mathbf{x})$$

Here the projection matrices $\mathbf{P}_i$ select out the coordinates corresponding to the sets B_i. That is, for $d \in E_i$, $e \in E$, $P_i[d,e] = [\![d = e]\!]$. Then by forming the Langragian we get the dual problem:

$$\max_{\boldsymbol{\lambda}_i} \min_{\mathbf{x},\mathbf{x}_i} \frac{1}{2}||\mathbf{x}||^2 + \langle \mathbf{x}, \mathbf{c}\rangle + \sum_i \tilde{f}_i(\mathbf{x}_i) + \boldsymbol{\lambda}_i^T(\mathbf{P}_i\mathbf{x} - \mathbf{x}_i)$$

$$\max_{\boldsymbol{\lambda}_i} -\frac{1}{2}||\mathbf{c} + \sum_i \mathbf{P}_i^T \boldsymbol{\lambda}_i||^2 - \sum_i \tilde{f}_i^*(\boldsymbol{\lambda}_i)$$

The dual functions $\tilde{f}_i^*$ are indicactor functions of the corresponding base polytopes B_{f_i}. We scale and concatenate the $\boldsymbol{\lambda}_i$ into a single vector $\mathbf{z}$ of size $N = \sum_{i=1}^{M} |E_i|$

$$\boldsymbol{\lambda}_i = \mathbf{S}_i \mathbf{Q}_i \mathbf{z} \in \mathbb{R}^n$$

$$\mathbf{R} = \sum_i \mathbf{P}_i^T \mathbf{S}_i \mathbf{Q}_i$$

The projection matrices $\mathbf{Q}_i$ select out the appropriate coordinates of $\mathbf{z}$ and we choose the scales $\mathbf{S}_i$ so that the matrix $\mathbf{R}$ is a projection $\mathbf{R}^2 = \mathbf{R}$. So then the problem can be written in a form with a single vector variable:

$$\min_{\mathbf{z} \in \mathbb{R}^N} \frac{1}{2}||\mathbf{c} + \mathbf{R}\mathbf{z}||^2 \tag{5.3.1}$$
$$\mathbf{S}_i \mathbf{Q}_i \mathbf{z} \in B_{f_i} \text{ for i} = 1 \ldots \text{M}$$

Note that any feasible $\mathbf{z}$ gives us a base vector for f: $\boldsymbol{\lambda} = \mathbf{c} + \mathbf{R}\mathbf{z} \in B_f$. We then apply accelerated gradient methods to this formulation to derive algorithms. However, in the implementations we maintain the vectors $\boldsymbol{\lambda}_i$ separately, because that is how we parallelize.

5.3.2 Outline of Algorithms

Note that Equation 5.3.1 is in a form to which we can simply apply standard first order methods to it. We implemented and tested three different first order methods:

1. Proximal Gradient
2. Accelerated Proximal Gradient (FISTA)
3. Proximal Gradient with Barzilai-Borwein stepsizes

The last technique simply applies proximal gradient, but uses aggressive stepsizes. The idea is that if L_k is an estimate of the Lipschitz constant used in the algorithm, $O(1/k^2)$ convergence is guaranteed if we ensure that our updates satisfy:

$$L_k \geq \frac{||\mathbf{R}(\mathbf{z}_{k+1} - \mathbf{z}_k)||^2}{||\mathbf{z}_{k+1} - \mathbf{z}_k||^2} \tag{5.3.2}$$

Here $\mathbf{R}$ is the hessian of our objective function. A conservative option to guarantee convergence is to backtrack whenever a constraint is violated; that is, discard the update $\mathbf{z}_{k+1}$, increase L_k and project again. It is then difficult to achieve a balance between making sure the stepsizes are large enough, but not to waste too many updates backtracking. Instead by using Barzilai–Borwein stepsize, we simply set L_{k+1} to be equal to quantity on the right-hand side of this Equation 5.3.2 for step k. The overall behavior is to take large steps when the projection is preventing progress in the objective, at the expense of ocasionally overstepping. This results in non-montonic convergence.

Accelerating the Min-Norm Algorithm. Can we use accelerated methods even when there is no algorithm for projection onto the submodular polytope? We outline a procedure that has worked well in some limited experiments. The problem is to maximize $-\|\sum \boldsymbol{\lambda}_i\|^2$ subject to $\boldsymbol{\lambda}_i \in B_{f_i}$. Initially let $\hat{\boldsymbol{\lambda}}_i^0 = \boldsymbol{\lambda}_i^0$.

1. $\mathbf{g}^k = -\sum_i \hat{\boldsymbol{\lambda}}_i^k$
2. Choose descent directions by maximizing the first order approximation of the objective with the greedy method: $\mathbf{y}_i^k = \arg\max_{\mathbf{y} \in B_{f_i}} (\mathbf{y} - \hat{\lambda}_i^k)^T \mathbf{g}^k$.
3. Find coefficients β_i^k through box-constrained least squares:

$$\min_{\beta_i \in [0,1]} \frac{1}{2} \left\| \sum_i \beta_i \mathbf{y}_i^k + (1-\beta_i)\hat{\lambda}_i^k \right\|^2 .$$

4. Update the vectors using an accelerated update rule: $\boldsymbol{\lambda}_i^{k+1} = \beta_i^k \mathbf{y}_i^k + (1-\beta_i^k)\hat{\lambda}_i^k$, $\hat{\boldsymbol{\lambda}}_i^{k+1} = \lambda_i^{k+1} + \frac{(\alpha_k - 1)}{\alpha_{k+1}}(\boldsymbol{\lambda}_i^{k+1} - \boldsymbol{\lambda}_i^k)$, $\alpha_{k+1} = \frac{1+\sqrt{1+4\alpha_k^2}}{2}$.
5. Repeat until convergence criterion is met.

In our experiments, we use another accelerated gradient method to find the β_i^k in step 3. Unfortunately, the vectors $\hat{\lambda}_i$ actually can leave the constraint set, although this appears to actually speed convergence.

5.4 Fast Proximal Threshold Potentials

The general principle of our consensus algorithms is to combine simple submodular functions for which projection onto the base polytope is efficient in order to project onto the base polytope of the sum. One class of functions for which projection is efficient are threshold potentials, introduced in Section 4.3. To review, a threshold potential is a positive modular function truncated at some treshold: $f(A) = \min(\tau, \mathbf{w}(A))$, where $\mathbf{w} \geq \mathbf{0}$ and $0 < \tau < \mathbf{w}(E)$. The base polytope for such a function is given by the intersection of a box with a plane: $B_f = \{\boldsymbol{\lambda} \in \mathbb{R}^n \mid \langle \mathbb{1}, \boldsymbol{\lambda} \rangle = \tau,\ \mathbf{0} \leq \boldsymbol{\lambda} \leq \mathbf{w}\}$. In this section, we outline two algorithms for the general problem of projecting onto a box-plane intersection (not necessarily a base polytope). The first method requires sorting the components of the vector to be projected, but is simple to implement. Our second method is a randomized algorithm that runs in expected linear time.

5.4.1 General Plane Intersection Projection

The following proposition will be useful for the characterization of projections onto convex subsets of planes. It suggests a general method for computing the projection of a single plane intersected with any convex set.

Proposition 5.5. *Let C be any closed convex set in $\mathbb{R}^n$, and P be the plane given by $\langle \mathbf{x}, \mathbf{a} \rangle = b$. Then, for $t \in \mathbb{R}$,*

$$\langle \Pi_C(\mathbf{x} + t\mathbf{a}), \mathbf{a} \rangle = b \Longleftrightarrow \Pi_C(\mathbf{x} + t\mathbf{a}) = \Pi_{C \cap P}(\mathbf{x}) \tag{5.4.1}$$

Proof. The backward direction of the proposition is obvious, since $\Pi_{C\cap P}(\mathbf{x}) \in P$. To prove the forward direction, we use the following facts:

- If C is any convex set, then $\mathbf{y} = \mathbf{x} - \Pi_C(\mathbf{x})$ implies that $\Pi_C(\mathbf{x} + \mathbf{y}) = \Pi_C(\mathbf{x})$.
- If P is a plane normal to the vector $\mathbf{a}$, then $\Pi_P(\mathbf{x} + \mathbf{a}) = \Pi_P(\mathbf{x})$.
- If C_1, C_2 are any convex sets, then $\mathbf{z} = \Pi_{C_1}(\mathbf{x} + \boldsymbol{\lambda}) = \Pi_{C_2}(\mathbf{x} - \boldsymbol{\lambda})$ implies that $\mathbf{z} = \Pi_{C_1 \cap C_2}(\mathbf{x})$. (This is a special case of Proposition 2.2.)

To prove the proposition we use $\boldsymbol{\lambda} = \mathbf{x} + t\mathbf{a} - \Pi_C(\mathbf{x} + t\mathbf{a})$, and by the above facts $\Pi_C(\mathbf{x} + \boldsymbol{\lambda}) = \Pi_P(\mathbf{x} - \boldsymbol{\lambda}) = \Pi_C(\mathbf{x} + t\mathbf{a}) = \Pi_{C\cap P}(\mathbf{x})$. ☐

How can we interpret the proposition? Suppose we start at a point $\mathbf{x}$, move in the direction normal to the plane for some distance, and then project onto C. If the resulting point is on the plane, then it must be the projection of $\mathbf{x}$ onto the intersection of the plane with C. Therefore to calculate the projection, it is a matter of finding this correct distance.

We conclude that if a convex set C is easy to project onto, this allows us to project almost as easily onto $C \cap P$. In some cases, this will give a closed form formula. For example, when C itself is an intersection of planes, then this reduces to the matrix inversion problem of underdetermined least squares. But if there is no closed form solution, then we can at least say that the Proposition reduces the problem of a multi-dimensional optimization problem to a 1-dimensional root finding problem. If we let $\rho(t) = \langle \Pi_C(\mathbf{x} + t\mathbf{a}), \mathbf{a} \rangle$, then we can use any numerical root-finding method to solve $\rho(t^*) = b$. Of course, in order for this to work, the projection Π_C must be computationally inexpensive to compute.

5.4.2 Box-Plane Projection Algorithms

For the particular case of Proposition 5.5 in which we are interested, the convex set C is a box: $\{\mathbf{x} \in \mathbb{R}^n \mid \boldsymbol{\ell} \leq \mathbf{x} \leq \mathbf{u}\}$. This is, of course, an extremely simple set to project onto. Each component of a vector to be projected need only be compared to the boundaries of the box and adjusted accordingly. That is, $\Pi_C(\mathbf{x}) = \max(\boldsymbol{\ell}, \min(\mathbf{u}, \mathbf{x}))$. As a consequence of the Proposition, we can project onto a box-plane intersection easily, provided that we can efficiently solve the following equation:

$$\langle \max(\boldsymbol{\ell}, \min(\mathbf{u}, \mathbf{x} + t^*\mathbf{a})), \mathbf{a} \rangle = b \tag{5.4.2}$$

Once the satisfying value t^* is known, the projected point is given by:

$$\Pi_{C \cap P}(\mathbf{x}) = \max(\boldsymbol{\ell}, \min(\mathbf{u}, \mathbf{x} + t^*\mathbf{a})) \tag{5.4.3}$$

Though we could use a general numerical method to find the root of Equation 5.4.2, we can do better with a routine specialized for this particular problem. In this case, the function $\rho(t)$ is monotonic and piecewise linear, which we can write as:

$$\rho(t) = -b + \langle \mathbf{a}, \max(\boldsymbol{\ell}, \min(\mathbf{u}, \mathbf{x} + t^*\mathbf{a})) \rangle$$

Algorithm 5.1: Projection onto a Box Intersecting a Plane

```
Input: Vectors $\boldsymbol{\ell}, \mathbf{u}, \mathbf{a}, \mathbf{x} \in \mathbb{R}^n$, scalar $b$, where
        $C := \{\mathbf{y} \in \mathbb{R}^n \mid \boldsymbol{\ell} \le \mathbf{y} \le \mathbf{u},\ \langle \mathbf{x}, \mathbf{a} \rangle = b\} \neq \emptyset$.
begin
    $P \leftarrow \emptyset$ ;
    for $i \leftarrow 1$ to $n$ do
        if $a[i] \neq 0$ and $\ell[i] < u[i]$ then
            $s \leftarrow a[i]^2 \operatorname{sign}(a[i])/2$;
            $t_\ell \leftarrow (\ell[i] - x[i])/a[i]$;
            $t_u \leftarrow (u[i] - x[i])/a[i]$;
            $P \leftarrow P \cup \{(s, t_\ell), (-s, t_u)\}$ ;
    $r \leftarrow -b + \langle \boldsymbol{\ell} + \mathbf{u}, \mathbf{a} \rangle / 2$ ;
    $t^* \leftarrow$ PLM Root$(0, r, P)$;
    $\mathbf{y}^* \leftarrow \max(\boldsymbol{\ell}, \min(\mathbf{u}, \mathbf{x} + t^* \mathbf{a}))$;
Output: $\mathbf{y}^* = \Pi_C(\mathbf{x})$.
```

$$= r + \sum_{(s_i, t_i) \in P} s_i |t - t_i|. \tag{5.4.4}$$

The formulas for the breakpoints t_i, slopes s_i, and offset r in terms of the problem data $(\boldsymbol{\ell}, \mathbf{u}, \mathbf{a}, b, \mathbf{x})$ are derived through simple algebraic manipulation; we thus list them in Algorithm 5.1 without further comment. Now the problem has been reduced to finding the root of a function of the form in Equation 5.4.4. There are at least two ways to handle this subproblem. Our first suggestion: sort the breakpoints t_i; then evaluate each value $\rho(t_i)$ in sequence until bracketing the root to an interval where ρ is linear; finally, solve the locally linear equation for the root t^*. This requires $O(n \log n)$ operations for the sort and $O(n)$ operations on the sorted data. The details of this procedure are listed as Algorithm 5.2. (Note that if we use Algorithm 5.2 within Algorithm 5.1, we can skip the first *for* loop and set $\tilde{q} \leftarrow 0$ and $\tilde{r} \leftarrow -b + \langle \min(\boldsymbol{\ell} \bullet \mathbf{a}, \mathbf{u} \bullet \mathbf{a}), \mathbb{1} \rangle$.)

Alternatively, we can use a randomized algorithm to find the root that runs in expected $O(n)$ operations. The earliest publication of this idea is in [PK90], but we were inspired by the more recent article [DSSSC08] for linear-time projection onto the ℓ_1 ball in a machine learning context. The principle behind this algorithm is the same as that of finding a quantile of a list of numbers in $O(n)$ operations. We randomly choose breakpoints as pivots, eliminating a fraction of the data from consideration at each iteration. We do this by simplifying the description of the input function in the range that the root must lie. That is, given input parameters q, r, s, t_i, we keep track of $\tilde{q}, \tilde{r}$ and a subset of indices $\tilde{I}$ such that for all t near

Algorithm 5.2: PLM Root: Sorted Version

```
Input: Piecewise Linear Monotonic Function $\rho(t) = qt + r + \sum_i s_i|t - t_i|$, represented
       by: Scalars $q, r$, List of ordered pairs $P = \{(s_i, t_i)\}_i$, sorted so that $t_i \le t_{i+1}$.
begin
    $\tilde{q} \leftarrow q$;    $\tilde{r} \leftarrow r$;
    for $i = 1$ to $|P|$ do
        $\tilde{q} \leftarrow \tilde{q} - s_i$ ;
        $\tilde{r} \leftarrow \tilde{r} + s_i t_i$ ;
    $t \leftarrow 0$;
    for $i = 1$ to $|P|$ do
        $\tilde{r} \leftarrow \tilde{r} + q(t_i - t)$;
        $t \leftarrow t_i$;
        if $\tilde{r} \ge 0$ then stop;
        ;
        $\tilde{q} \leftarrow \tilde{q} + 2s_i$ ;
    if $\tilde{q} \ne 0$ then
        $t \leftarrow t - \tilde{r}/\tilde{q}$;
Output: $t$ satisfying $\rho(t) = 0$
```

the root of the function we have:

$$qt + r + \sum_{i \in I} s_i|t - t_i| = \tilde{q}t + \tilde{r} + \sum_{i \in \tilde{I}} s_i|t - t_i|$$

The details of this are shown in Algorithm 5.3.

5.5 Experiments

Since we can easily compute the proximal operator for a threshold potential, we use our algorithms to minimize a sum of threshold potentials. We tested on a synthetic problem designed to mimic semi-supervised learning. Given n points in some metric space, we define weight vectors: $w[j] = w_0 \exp(-d(i,j)/d_0)^2)$ where $d(i,j)$ is the distance between i and j, and w_0 and d_0 are model parameters. Then given a vector of partial (possibly noisy) labels $\mathbf{c} \in \{-1, 0, +1\}$, we minimize the following to classify the points:

$$f(A) = \mathbf{c}(A) + \sum_i \min(\mathbf{w}_i(A), \mathbf{w}_i(E \setminus A))$$

See Figure 5.1 for an example of a problem. Figure 5.4 and Figure 5.3 show examples of program output using two of our techniques (FISTA and the Barzilai–Borwein proximal

Algorithm 5.3: PLM Root: Unsorted Version

```
Input: Piecewise Linear Monotonic Function $\rho(t) = qt + r + \sum_i s_i|t - t_i|$, represented
       by: Scalars $q, r$, Unsorted list of ordered pairs $P = \{(s_i, t_i)\}_i$.
begin
    $\tilde{q} \leftarrow q$;   $\tilde{r} \leftarrow r$;   $\tilde{P} \leftarrow P$;
    while $\tilde{P} \neq \emptyset$ do
        Choose $(s_0, t_0)$ at random from $\tilde{P}$ ;
        $dq_\ell \leftarrow 0$;   $dr_\ell \leftarrow 0$;   $P_\ell \leftarrow \emptyset$;
        $dq_u \leftarrow 0$;   $dr_u \leftarrow 0$;   $P_u \leftarrow \emptyset$;
        for $(s_i, t_i) \in \tilde{P}$ do
            if $t_i > t_0$ then
                $P_u \leftarrow P_u \cup \{(s_i, t_i)\}$;
            else
                $dq_u \leftarrow dq_u + s_i$;
                $dr_u \leftarrow dr_u - s_i t_i$;
            if $t_i < t_0$ then
                $P_\ell \leftarrow P_\ell \cup \{(s_i, t_i)\}$;
            else
                $dq_\ell \leftarrow dq_\ell - s_i$;
                $dr_\ell \leftarrow dr_\ell + s_i t_i$;
        $v_0 \leftarrow (\tilde{q} + dq_u + dq_\ell)t_0 + \tilde{r} + dr_u + dr_\ell$ ;
        if $v_0 < 0$ then
            $\tilde{q} \leftarrow \tilde{q} + dq_u$;
            $\tilde{r} \leftarrow \tilde{r} + dr_u$;
            $\tilde{P} \leftarrow P_u$;
        else if $v_0 > 0$ then
            $\tilde{q} \leftarrow \tilde{q} + dq_\ell$;
            $\tilde{r} \leftarrow \tilde{r} + dr_\ell$;
            $\tilde{P} \leftarrow P_\ell$;
        else
            $t \leftarrow t_0$;
            stop while;
    if $\tilde{P} = \emptyset$ then
        $t \leftarrow -\tilde{r}/\tilde{q}$ ;
Output: $t$ satisfying $\rho(t) = 0$
```

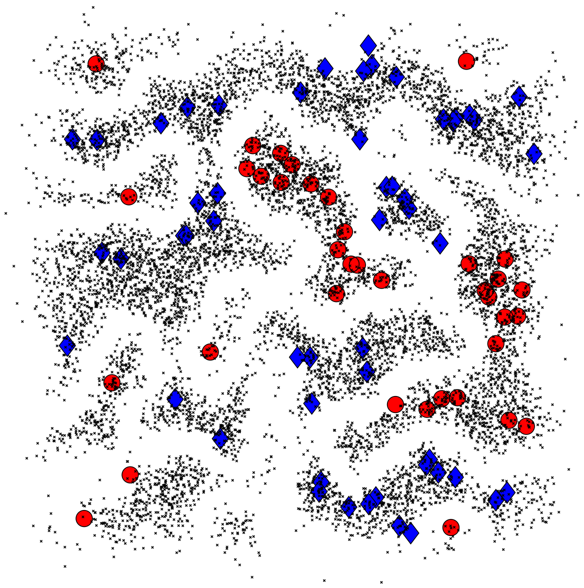


Figure 5.1: Example Synthetic Semi-Supervised Learning Problem. 10000 total points. 90 labeled points.

gradient). Note the nonmonotonic convergence of the latter method. In Figure 5.2, we show the results dividing up the number of potentials across different simulated processors to examine the effect of parallelization. Since the main bottleneck of implementation is the computing the projections, which parallelizes perfectly across processors, we see close to ideal speedup times.

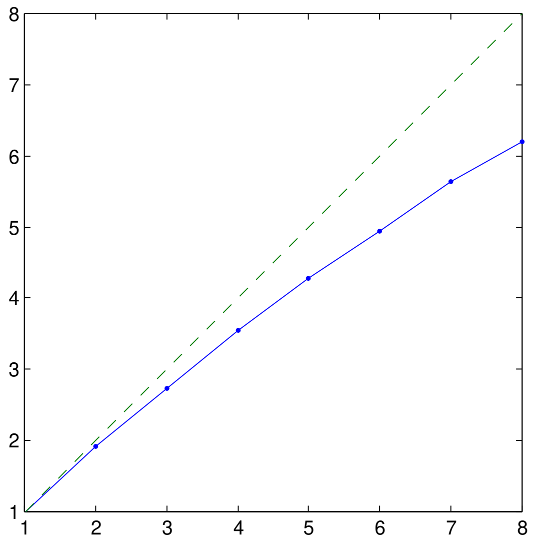


Figure 5.2: Speedup on example problem with 10000 total points using Proximal Gradient with Barzilai–Borwein stepsizes,

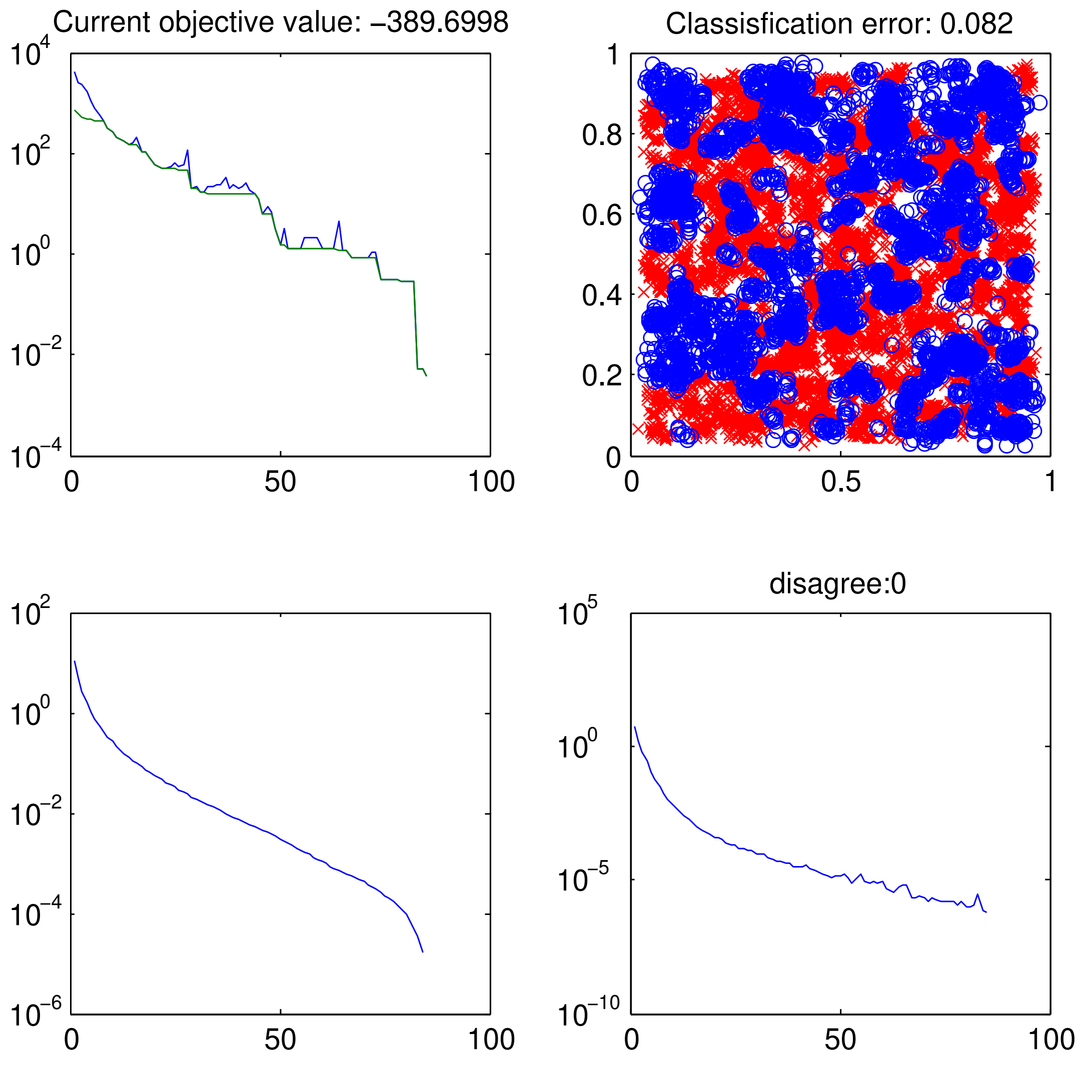


Figure 5.3: Convergence with FISTA method. a) Duality Gap b) Resulting Classification c) Norm of λ, d) Residual

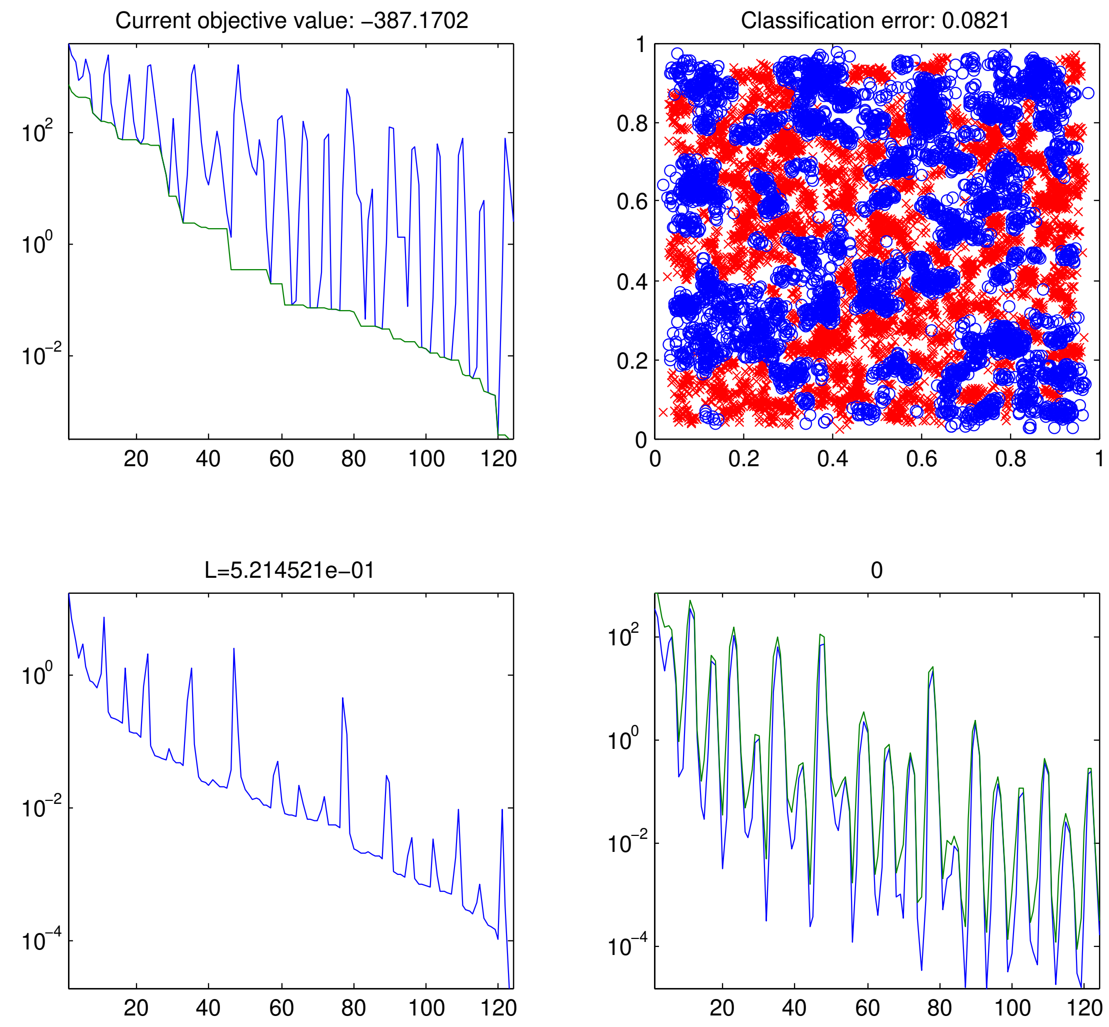


Figure 5.4: Convergence with Barzilai–Borwein method. a) Duality gap b) Resulting Classification c) Norm of λ, d) Residual

Chapter 6

Learning Fourier Sparse Set Functions

6.1 Introduction

Suppose we wish to sketch the evolution of a massive network: we are given a sequence of networks, where between each step, only few edges get added or removed. Can we compute a small number of statistics, which allow, in hindsight, to reconstruct which edges got added or removed, without storing the entire network? In this chapter, we show that this is possible by observing and storing a small number of random cuts and their values.

Formally, we consider the problem of learning a set function f (mapping subsets of some finite set E of size n to the real numbers) by observing its value on a few sets. Without observing any structure, we clearly need an exponential (in n) number of observations to approximate the function well over all sets, so we need an appropriate regularity condition. To that end, we consider the situation where f is *smooth* in the sense of having a decaying Fourier (Hadamard-Walsh) spectrum. One natural example of this is the cut function of a (possibly directed) graph, or generalized additively independent (GAI) functions [Fis67], that decompose into a sum of local terms.

By leveraging recent results from sparse recovery [Ver12], we show that if the function is sparse in the Fourier domain, having at most k nonzero coefficients, and support contained in a known collection $\mathcal{P}$ of size p, then it is possible to efficiently recover the function exactly from very few samples. In particular, suppose we pick $O(k \log^4(p))$ sets uniformly at random. Then with very high probability (over this random choice), observing the values of the function on these sets is sufficient to *exactly* reconstruct it.

Besides decaying Fourier spectrum, many set functions encountered in practice satisfy additional properties. In particular, we consider *submodular* functions, which form a natural discrete analogue of convex functions [Lov83]. Submodularity is satisfied by numerous set functions encountered in practice, such as the cut function in graphs [Sch03], entropy [KK80], mutual information [KSG08] etc. The problem of learning submodular functions has received considerable attention recently [GHIM09, BH11]. However, approximating a submodular function by a factor better than $\sqrt{n}/\log n$ uniformly over all sets requires an *exponential number* of function evaluations, even if those can be adaptively chosen [GHIM09]. We show that submodularity implies certain structure in the Fourier domain, which can be exploited to reduce the number of required samples even further.

Besides allowing us to *sketch the evolution of large graphs* by observing the value of a few random cuts, as mentioned above, our results show that practically relevant set functions, such as certain *valuation functions*, a fundamental concept in economics capturing substitutability of certain products, can be efficiently learned from few examples. Another natural application is in speeding up submodular optimization: standard algorithms assume that the function f is presented by an *oracle*, which evaluates f on any set. In general, evaluating f can be very costly (requiring the solution of a large linear system, or perform large-scale simulations). In such a setting, if f is Fourier-sparse, we can approximate it compactly using a small number of random sets, and then optimize the compact representation instead.

In summary, our main contributions are:

- We show that it is possible to learn Fourier k-sparse set functions exactly using $O(k\log^4(p))$ random samples. This reconstruction is robust to noise.
- We show that properties such as symmetry and submodularity of f imply structure in the Fourier domain, which can be exploited to obtain further reduction in sample complexity.
- We demonstrate our algorithm on a problem of sketching the evolution of a graph, and on approximate submodular optimization.

6.2 Background

Recall that we define $\mathcal{H} = \mathbb{R}^{2^E}$ to be the space of set functions with ground set E, which is of size n. Suppose we are given the value of a set function $f \in \mathcal{H}$ on some collection of subsets. For now, let us assume that these observations are noise free – we will relax this condition later. Under what conditions can we hope to recover f? Clearly, without any assumptions about f, we need an exponential number (in n) of samples in order to obtain exact reconstruction. However, if f is smooth in some way, we may hope to do better. Similar to continuous functions, a natural smoothness condition is decaying Fourier spectrum.

6.2.1 The Fourier transform on set functions.

Set functions can equivalently be represented as real-valued functions of boolean vectors, known as pseudoboolean functions. Just as the set of boolean vectors $\{0,1\}^n$ forms the commutative group $\mathbb{Z}_2^n$ under addition modulo 2, the power set 2^E forms an equivalent group under the operation of symmetric set difference: $A \ominus B := (A \setminus B) \cup (B \setminus A)$. So the space $\mathcal{H}$ has a natural Fourier (also called Hadamard-Walsh) basis, and in our set function notation the corresponding Fourier basis vectors, or Fourier modes, are defined as:

$$\psi_B(A) := (-1)^{|A \cap B|}.$$

The space $\mathcal{H}$ is endowed with the standard inner product: $\langle f, g\rangle := 2^{-n} \sum_{A \in 2^E} f(A)g(A)$. The Fourier transform of a function is given by its inner products with Fourier modes:

$$\widehat{f}(B) := \langle f, \psi_B \rangle = 2^{-n} \sum_{A \in 2^E} f(A)(-1)^{|A \cap B|}.$$

Note that the sum in this definition has exponentially many terms, so it is not practical to evaluate directly. As with any orthonormal basis, we have a reconstruction formula: $f(A) = \sum_{B \in 2^E} \widehat{f}(B)\psi_B(A)$.

The Fourier support of a set function is the collection of subsets with nonzero Fourier coefficient: $\mathrm{Supp}[\widehat{f}] := \{B \in 2^E \mid \widehat{f}(B) \neq 0\}$. Given a collection of subsets $\mathcal{P} \subseteq 2^E$, let $\mathcal{H}_{\mathcal{P}} := \{f \in \mathcal{H} \mid \mathrm{Supp}[\widehat{f}] \subseteq \mathcal{P}\}$ be the subspace with Fourier support contained in $\mathcal{P}$. We assume we have some a priori knowledge about the Fourier support which gives a natural choice for $\mathcal{P}$. We discuss this in further detail in Section 6.4, but for now assume $\mathcal{P}$ is some known collection of polynomial size. One illustrative example is the collection of sets of size q or less: $\mathcal{P}_q := \{B \in 2^E \mid |B| \leq q\}$. As it is particularly important, we denote this function space, consisting of all functions of order q or less, by the symbol $\mathcal{H}_q$. The number of free parameters is $p = \sum_{j=0}^{q} \binom{n}{j}$, which is not too large when $q = 2$.

Now with $\mathcal{P}$ fixed, suppose we restrict ourselves to $f \in \mathcal{H}_{\mathcal{P}}$. Can we recover f with a subexponential number of samples? In the next section, we show that if the Fourier support is small, then this is indeed possible by leveraging recent results from sparse reconstruction.

6.3 Conditions for Recovery

Since a set function is uniquely determined by its Fourier transform, recovering a Fourier-sparse function can be thought of as recovery of a sparse vector in $\mathbb{R}^{2^n}$. For large n, even representing such vectors will be intractable. However, if we know that the Fourier support of a function is contained in $\mathcal{P}$, then, instead, we treat $\widehat{f}$ as a sparse vector in $\mathbb{R}^p$. We will show that it is possible to uniquely recover *any* $f \in \mathcal{H}_{\mathcal{P}}$ with $|\operatorname{Supp}[\widehat{f}]| \leq k$ by observing the values $f_{\mathcal{M}}$ (with high probability over the choice of measurement sets $\mathcal{M}$), provided that

$$m = O(k \log^4(p)).$$

Matrix vector notation. In the problems that we consider, we observe the function f evaluated on sets from a measurement collection $\mathcal{M} = \{A_i\}$ of size m. We arrange these measurements in a vector $\mathbf{f}_{\mathcal{M}} \in \mathbb{R}^m$, where $\mathbf{f}_{\mathcal{M}}[i] := f(A_i)$ for $i = 1 \ldots m$. Note the bold typeface used to distinguish vectors from set functions. Furthermore, we will assume that the Fourier support is contained in a known potential support collection $\mathcal{P} = \{B_j\}$ of size p. We denote $\hat{\mathbf{f}}_{\mathcal{P}} \in \mathbb{R}^p$ for the the corresponding vector of Fourier coefficients, where $\hat{\mathbf{f}}_{\mathcal{P}}[j] := \widehat{f}(B_j)$ for $j = 1 \ldots p$. Lastly, we denote $\mathbf{\Psi}_{\mathcal{M},\mathcal{P}}$ for the $m \times p$ matrix which relates the two vectors,

$$\mathbf{\Psi}_{\mathcal{M},\mathcal{P}}[i,j] := \psi_{B_j}(A_i) = (-1)^{|A_i \cap B_j|}. \tag{6.3.1}$$

Then for $f \in \mathcal{H}_{\mathcal{P}}$ we have:

$$\mathbf{f}_{\mathcal{M}} = \mathbf{\Psi}_{\mathcal{M},\mathcal{P}} \hat{\mathbf{f}}_{\mathcal{P}}. \tag{6.3.2}$$

So recovery of f is equivalent to recovery of a sparse vector from linear measurements.

Restricted Isometry. The problem of finding a k-sparse vector from an underdetermined linear system has received significant attention in the context of *compressive sensing* [CRT06, Don06]. A sufficient condition for recovery is that the sensing matrix satisfies a key property, the *Restricted Isometry Property (RIP)*. In order to ensure that our measurement matrix $\mathbf{\Psi}_{\mathcal{M},\mathcal{P}}$ satisfies this property, we simply choose the measurement sets $\mathcal{M} = \{A_1, \ldots, A_m\}$ uniformly at random. Then, as we will see below, results in random matrix theory imply that with high probability (for any fixed $\mathcal{P}$), the measurement matrix indeed satisfies RIP. This insight opens up a vast collection of tools from compressive sensing for the purpose of recovering set functions.

The idea behind RIP is that a matrix acts approximately as an isometry on sparse vectors. Specifically, suppose for some sparsity level k and $\delta \geq 0$ we have:

$$(1-\delta)\|\mathbf{x}\|^2 \leq \|\mathbf{\Phi x}\|^2 \leq (1+\delta)\|\mathbf{x}\|^2 \text{ for all } \mathbf{x} \in \mathbb{R}^p \text{ with } \operatorname{Supp}[\mathbf{x}] \leq k \tag{6.3.3}$$

We define $\delta_k(\mathbf{\Phi})$ as the smallest value of δ such that Equation 6.3.3 holds. An easy consequence of this definition is that the linear measurement vector $\mathbf{y} = \mathbf{\Phi x}_0$ uniquely determines any k-sparse $\mathbf{x}_0$ if and only if $\delta_{2k} < 1$. Furthermore, with a stronger assumption on the isometry constants, the original vector can be recovered by solving a convex optimization problem:

$$\min_{\mathbf{x} \in \mathbb{R}^p} \|\mathbf{x}\|_1, \quad \mathbf{\Phi x} = \mathbf{y} \tag{6.3.4}$$

Originally, [CT05] showed that Equation 6.3.4 recovers any k-sparse $\mathbf{x}_0$ if $\delta_{3k} + 3\delta_{4k} < 2$, but this condition has been weakened several times, most recently in [Fou10], which gives the condition $\delta_{2k}(\Phi) < 3/(4+\sqrt{6}) \approx .465$. Furthermore, as discussed below, this result can be generalized to noisy measurements.

Main Reconstruction Result As discussed above, RIP is a very powerful property, but it is not easy to check that any given matrix satisifies it. In fact, most constructions are based on choosing measurements with randomness and then calculating the likelihood of RIP. Perhaps the simplest such case is for random matrices with independent subgaussian entries. However, in our case, we are randomly sampling rows from an orthonormal matrix with bounded entries. Fortunately, as shown Rudelson and Vershynin [RV08] [Ver12], even in this setting, as long as $m = O(k \log^4(p))$, the expectation of the kth RIP constant is small. More recently, [Rau10] demonstrated that RIP for such matrices holds with high probability. Our result below is essentially Theorem 4.4 of [Rau10] as applied to our case of set functions.

Theorem 6.1. *For a fixed collection $\mathcal{P} = \{B_j\}_{j=1}^p \subset 2^E$, suppose a measurement collection $\mathcal{M} = \{A_i\}_{i=1}^m \subset 2^E$ is chosen by selecting the sets A_i uniformly at random. Define the matrix $\mathbf{\Psi}_{\mathcal{M},\mathcal{P}} \in \mathbb{R}^{m \times p}$ as in Equation 6.3.1. Then there exist universal constants $C_1, C_2 > 0$ such that if $k \leq p/2$, and $m \geq \max(C_1 k \log^4(p), C_2 k \log(1/\delta))$, except for an event of probability no more than δ, the following holds for* all *$f \in \mathcal{H}_{\mathcal{P}}$:*

For any noise level $\eta \geq 0$ and any noisy vector of measurements $\mathbf{y} \in \mathbb{R}^m$ within that noise level: $\|\mathbf{y} - \mathbf{f}_{\mathcal{M}}\|_2 \leq \eta$, suppose $g \in \mathcal{H}_{\mathcal{P}}$ has Fourier transform vector $\hat{\mathbf{g}}_{\mathcal{P}} \in \mathbb{R}^p$ given by :

$$\hat{\mathbf{g}}_{\mathcal{P}} = \arg\min_{\mathbf{x} \in \mathbb{R}^p} \|\mathbf{x}\|_1, \ \|\mathbf{y} - \mathbf{\Psi}_{\mathcal{M},\mathcal{P}} \mathbf{x}\|_2 \leq \eta. \tag{6.3.5}$$

Then the following bound holds for some universal constants C_3, C_4:

$$\|f - g\|_2 \leq \frac{C_3}{\sqrt{k}} \mu_k(\hat{\mathbf{f}}) + \frac{C_4}{\sqrt{m}} \eta, \tag{6.3.6}$$

where the quantity $\mu_k(\cdot)$ is defined as the ℓ_1 error of the best k-sparse approximation.

$$\mu_k(\mathbf{x}) := \min_{\text{Supp}(\mathbf{z}) \leq k} \|\mathbf{x} - \mathbf{z}\|_1 \tag{6.3.7}$$

In particular, if $\hat{\mathbf{f}}_{\mathcal{P}}$ is k-sparse and $\eta = 0$, then $g = f$.

Therefore, we obtain a strong guarantee for efficiently (using convex optimization) learning Fourier-sparse set functions, robust against measurement noise. Note that, up to log factors, this matches lower bounds of [CJK11], who show that $\Omega(k \log n)$ measurements are necessary for recovery of a k sparse function in $\mathcal{H}_q$ with q fixed.

6.4 Classes of Set Functions

In general, p is superlinear in n, so even though Equation 6.3.4 is equivalent to a Linear Program, it will not necessarily lead to an efficient recovery algorithm. In the extreme case, if $\mathcal{P} = 2^E$, then even calculating a single matrix-vector product $\mathbf{\Psi}^T_{\mathcal{M},\mathcal{P}}\mathbf{y}$ is difficult. So even though the recovery guarantees of Theorem 6.1 apply to arbitrary collections $\mathcal{P}$, we need to make some further assumptions about our function to get a practical algorithm for recovery.

6.4.1 Symmetric functions.

One natural structural property, obeyed by set functions commonly arising in practice, is *symmetry*. That is, for all sets A it holds that $f(A) = f(E \setminus A)$. Examples of functions satisfying this property are the cut function in undirected graphs, as well as the mutual information, both considered in our experiments (cf. Section 6.6). It turns out that symmetry already implies interesting structure in the Fourier domain:

Proposition 6.2. *Let f be a symmetric set function. Then for all sets B of odd cardinality, it holds that $\widehat{f}(B) = 0$.*

Therefore, symmetry already implies that we can restrict $\mathcal{P}$ only to sets of odd cardinality.

6.4.2 Low order functions.

In 3.2.2, we defined the subspace $\mathcal{H}_q$ of order q functions. The following characterizations are all equivalent:

Properties 6.1. Low Order Characterizations

1. $\Delta_B f(A) = 0$ for all $A, B \in 2^E$ with $|B| > q$.
2. $\widehat{f}(B) = 0$ for all $B \in 2^E$ with $|B| > q$.
3. $f(A) = \sum_B [\![B \subseteq A,\ |B| \leq q]\!] \Delta_B f(\emptyset)$ for all $A \in 2^E$.
4. $\sum_{\substack{J \subseteq \{0,1\ldots q+1\} \\ 0 \in J}} (-1)^{|J|} f\left(\ominus_{i \in J} A_i\right) = 0,$ for all $A_0, A_1 \ldots A_{q+1} \in 2^E$.
5. There exist $g_i \in \mathcal{H}$ and $|B_i| \leq q$ such that $f(A) = \sum_i g_i(A \cap B_i)$ for all $A \in 2^E$.

Many set functions f are low-order, or well-approximated by a low-order function. Recovery of an order 1 function is equivalent to classical compressed sensing with a Bernoulli measurement matrix (assuming we ignore the constant offset $f(\emptyset)$). Recovery of a symmetric order 2 function can be thought of as recovering a graph from values of a cut function, a problem which received considerable interest, partly due to several problems arising in computational biology [ABK+04, GK00, CK10]. We can see the correspondence as follows: given a weighted undirected graph, defined by a matrix of edge weights $\mathbf{W}$, define the symmetric cut function over sets of vertices of the graph:

$$\phi_W(A) := \sum_{u \in A, v \notin A} W[u,v]. \tag{6.4.1}$$

Then the Fourier transform can be computed explicity:

$$\widehat{\phi}_W(B) = \begin{cases} \frac{1}{2}\sum_{u,v} W[u,v], & B = \emptyset \\ -\frac{1}{2}W[u,v], & B = \{u,v\} \\ 0, & \text{otherwise.} \end{cases} \tag{6.4.2}$$

Hence there is a simple linear correspondence between weights of a graph and the 2nd order Fourier coefficients of $\phi_{\mathbf{W}}$. Clearly, this correspondence works in reverse, i.e., given any symmetric 2nd order function f, there is a unique graph such that $f(A) - f(\emptyset) = \phi_{\mathbf{W}}(A)$. In the general case, functions in $\mathcal{H}_q$ can be thought of as cut functions of hypergraphs of degree q, as considered in [BM10].

6.4.3 Submodular functions.

Another structural property exhibited by many set functions of practical importance is *submodularity*, a natural discrete analogue of convexity [Lov83]. We present a definition of submodular functions emphasizing the Fourier transform. The shift operator and discrete difference operator are eigenfunctions of the Fourier transform. The shift operator applied to a Fourier mode is $S_B\psi_C(A) = \psi_C(B)\psi_C(A)$, and the discrete difference of a Fourier mode is $\Delta_B\psi_C(A) = [\![B \subseteq C]\!]\,(-2)^{|B|}\psi_C(A)$. Submodular functions are those with nonnegative second order differences: $\Delta_B f(A) \leq 0$ where A and B are disjoint, and B is a set of size two. We can drop the restriction that A and B are disjoint by multiplying by the corresponding Fourier mode, giving us following characterization of the cone of submodular functions:

$$\mathcal{H}_2^- = \{ f \in \mathcal{H} \mid \psi_B(A)\Delta_B f(A) \leq 0 \text{ for all } A, B \in 2^E,\ |B| = 2 \}.$$

While submodularity does not immediately restrict the set $\mathcal{P}$ of candidate supports, it immediately implies dependence among the Fourier coefficients, (a subset of) which can be encoded as constraints in the convex program solved during recovery. In particular, submodularity can be *characterized* in the Fourier domain. By using Equation 3.2.5 to express a second order difference as a sum of Fourier coefficient, we see that f is submodular if and only if for all $|B| = 2$ and $A \subseteq E \setminus B$:

$$\widehat{f}(B) + \sum_{C \in 2^E} [\![B \subsetneq C]\!] \widehat{f}(C) \psi_C(A) \leq 0 \tag{6.4.3}$$

Checking submodularity is no easier in the Fourier domain; it still requires checking at least $2^{n-2}\binom{n}{2}$ inequalities. However, we can get a necessary condition for submodularity.

Proposition 6.3. *For all $f \in \mathcal{H}_2^-$, and $|B| = 2$, $B \subsetneq C$,*

$$\widehat{f}(B) + |\widehat{f}(C)| \leq 0. \tag{6.4.4}$$

Proof. Given a particular C_1 such that $B \subsetneq C_1$, let $\mathcal{Q} := \{A \in 2^E \mid \psi_{C_1}(A) = \operatorname{sign}(\widehat{f}(C_1))\}$. By summing Equation 6.4.3 over all $A \in \mathcal{Q}$, we have:

$$\sum_{A \in \mathcal{Q}} \Big(\widehat{f}(B) + \sum_{C \in 2^E} [\![B \subsetneq C]\!] \widehat{f}(C) \psi_C(A) \Big) \leq 0$$

$$|\mathcal{Q}|\, \widehat{f}(B) + \sum_{C \in 2^E} [\![B \subsetneq C]\!] \widehat{f}(C) \sum_{A \in \mathcal{Q}} \psi_C(A) \leq 0 \tag{6.4.5}$$

Fix a set C_2 from the inner sum of Equation 6.4.5. We claim if $C_1 \neq C_2$, then there is some set D such that $A \in \mathcal{Q} \Leftrightarrow A \ominus D \in \mathcal{Q}$, and $\psi_{C_2}(A) = -\psi_{C_2}(A \ominus D)$, so $\sum_{A \in \mathcal{Q}} \psi_{C_2}(A) = 0$ by symmetry. To construct a set D which will satisfy the claim, at least one of the following two possibilities must work:

1. Choose any $c \in C_2 \setminus C_1$, and let $D = \{c\}$
2. Choose any $b \in B$, and $c \in C_1 \setminus C_2$, and let $D = \{b, c\}$

In summary, we have:

$$\sum_{A \in \mathcal{Q}} \psi_{C_2}(A) = \begin{cases} 0 & \text{if } B \subseteq C_2 \text{ and } C_2 \neq C_1 \\ |\mathcal{Q}| \operatorname{sign}(\widehat{f}(C_1)) & \text{if } C_2 = C_1 \end{cases}$$

Thus Equation 6.4.5 reduces to $|\mathcal{Q}|\widehat{f}(B)+\widehat{f}(C_1)|\mathcal{Q}|\operatorname{sign}(\widehat{f}(C_1)) \leq 0$, which is then equivalent to Equation 6.4.4 as desired. □

This has an immediate simple implication about the support of a submodular function:

Corollary 6.4. *For $f \in \mathcal{H}_2^-$, if $C \in \operatorname{Supp}[\widehat{f}]$, then $B \in \operatorname{Supp}[\widehat{f}]$ for all $B \subset C$ with $|B| = 2$.*

Besides providing some intuition about the Fourier support of submodular functions, Equation 6.4.4 gives a relatively simple convex constraint that can be incorporated into our recovery program. In general, adding any valid convex constraint can never increase our recovery error (a simple consequnce of convexity), and in practice it often decreases it.

There is another such useful constraint for any function which is low order in addition to being submodular. We can fully characterize third order submodular functions in terms of $\binom{n}{2}$ inequalities.

Proposition 6.5. *For all $f \in \mathcal{H}_3$, then $f \in \mathcal{H}_2^-$ if and only if for all $|B| = 2$:*

$$\widehat{f}(B) + \sum_{e \in E \setminus B} |\widehat{f}(B+e)| \leq 0. \tag{6.4.6}$$

Proof. For third order submodular functions, Equation 6.4.3 reduces to the following, which must hold for all $|B| = 2$ and $A \subseteq E \setminus B$:

$$\widehat{f}(B) + \sum_{e \in E \setminus B} \widehat{f}(B+e)\psi_{B+e}(A) \leq 0 \tag{6.4.7}$$

To show that Equation 6.4.6 is necessary for submodularity, apply Equation 6.4.7 to the set $A^* = \{ e \in E \setminus B \mid \widehat{f}(B+e) < 0 \}$. By construction, we have:

$$\widehat{f}(B+e)\psi_{B+e}(A^*) = |\widehat{f}(B+e)|$$

And this implies Equation 6.4.6. For the converse statement, note that A^* is the choice of A that maximizes the left-hand side of Equation 6.4.7. That is, for all $A \subseteq E \setminus B$, we have $\widehat{f}(B+e)\psi_{B+e}(A) \leq |\widehat{f}(B+e)|$, which means that Equation 6.4.6 implies Equation 6.4.7. □

Similarly, for fourth order submodular functions, we can get a necessary but not sufficient condition which is stronger than Equation 6.4.4. For all $f \in \mathcal{H}_4 \cap \mathcal{H}_2^-$ and $|B| = 2$, we have:

$$\widehat{f}(B) + |\widehat{f}(B+s) \pm \widehat{f}(B+t)| \pm \widehat{f}(B+s+t) \leq 0$$

The problem of recovering a submodular function has been studied in [CKKL12] through noise stability. One consequence of this property is that for any submodular function we have:

$$\sum_{B \in 2^E} [\![|B| \geq 2,\ |B| \text{ even}]\!]\, \widehat{f}(B) + \left(\sum_{B \in 2^E} [\![|B| \geq 2]\!]\, (|B|-1)\, \widehat{f}(B)^2 \right)^{1/2} \leq 0$$

6.5 Reconstruction Algorithms

In Section 6.3, we have shown that the problem of learning Fourier-sparse set functions can be reduced to the Compressed Sensing paradigm of recovery of a sparse vector from RIP measurements. This insight allows us to open up a cornucopia of algorithms that have been developed for this setup [TW10]. In particular, several greedy algorithms such as Orthogonal Matching Pursuit can explicitly take advantage of RIP to guarantee recovery, as shown in [Tro04]. For our experiments, we take the approach of convex optimization. Rather than solving Equation 6.3.4 exactly, we minimize the Lagrangian formulation so that we can apply an accelerated proximal method such as that of [AT06],

$$\min_{\mathbf{x} \in \mathbb{R}^p} \|\mathbf{x}\|_1 + \frac{1}{2\mu} \|\mathbf{\Psi}_{\mathcal{M},\mathcal{P}} \mathbf{x} - \mathbf{y}\|^2. \tag{6.5.1}$$

In our experiments, we use the toolbox TFOCS [BCG11], which requires only that we supply a method to apply $\mathbf{\Psi}_{\mathcal{M},\mathcal{P}}$ and $\mathbf{\Psi}_{\mathcal{M},\mathcal{P}}^T$. In the case of second order set functions, we do not need to store the entire $m \times p$ matrix, and there is a formula that only requires $O(mn)$ storage. Let $\mathbf{\Psi}_{\mathcal{M},q} := \mathbf{\Psi}_{\mathcal{M},\mathcal{P}_q}$ be the subsampled $m \times \binom{n}{q}$ Fourier matrix where the columns correspond to the sets of size q. So the matrix $\mathbf{\Psi}_{\mathcal{M},1} \in \mathbb{R}^{m \times n}$ is given by the formula: $\mathbf{\Psi}_{\mathcal{M},1}[i,j] = 1 - 2[\![j \in A_i]\!]$. If the 2nd order Fourier coeffients from $\mathbf{x} \in \mathbb{R}^{\binom{n}{2}}$ are arranged in the off-diagonal elements of an $n \times n$ a matrix $\mathbf{X}$, then the elements of $\mathbf{\Psi}_{\mathcal{M},2}\mathbf{x}$ are the diagonal elements of $\mathbf{\Psi}_{\mathcal{M},1}\mathbf{X}\mathbf{\Psi}_{\mathcal{M},1}^T$, and the transpose operation is $\mathbf{\Psi}_{\mathcal{M},2}^T\mathbf{r} = \mathbf{\Psi}_{\mathcal{M},1}^T \operatorname{diag}(\mathbf{r})\mathbf{\Psi}_{\mathcal{M},1}$.

6.5.1 Exploiting structure in the Fourier domain.

In Section 6.4, we have shown that submodularity implies constraints about the relative magnitudes of the Fourier coefficients. In addition to encoding this structure into the convex program to improve recovery, this structure can further be exploited to extend our technique to higher order functions (where the collection $\mathcal{P}$ can become intractably large). The key step in most sparse recovery algorithms is to find the largest magnitude elements of $\mathbf{\Psi}^T_{\mathcal{M},\mathcal{P}}\mathbf{r}$ given a residual vector $\mathbf{r}$. For example, first order methods applied to Equation 6.5.1, such as the ones we use, are equivalent to iterative soft-thresholding. While we could find the largest magnitude elements of $\mathbf{\Psi}^T_{\mathcal{M},\mathcal{P}}\mathbf{r}$ by simply applying the full transformation and sorting, one can use submodularity to avoid having to compute the entire set of higher-order coefficients. For example, if the function is 3rd order and submodular, we can apply Equation 6.4.6, and note for $|B| = 3$,

$$|\widehat{f}(B)| \leq \min_{b \in B} -\widehat{f}(B-b) - \sum_{a \in E \setminus B} |\widehat{f}(B-b+a)|$$

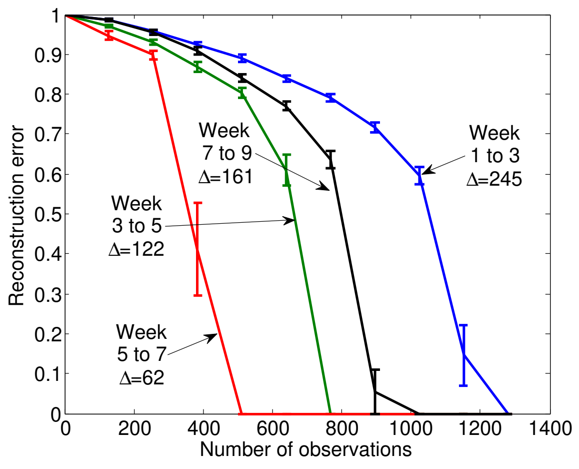
So these constraints can be used to speed up the identification of the largest magnitude coefficients, as we need only compute the 3rd order coefficients with sufficient slack. We leave a detailed investigation of this direction open for future work.

6.6 Applications and Experiments

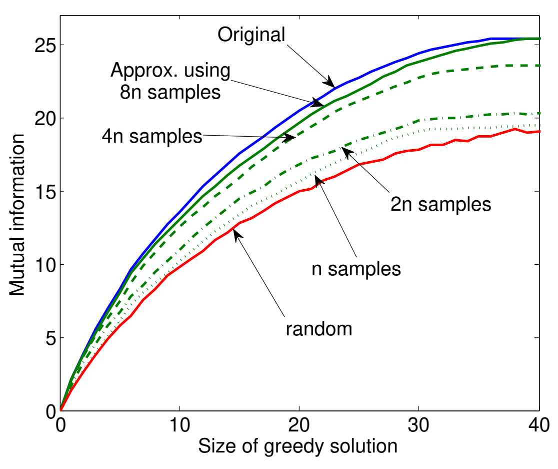
We evaluate our approach towards learning set functions on two real-world data sets. We also use synthetic data to demonstrate our claim that enforcing submodularity through convex constraints can improve recovery of submodular functions.

6.6.1 Sketching graph evolution

We consider the problem of reconstructing (differences between) graphs by observing random cuts. Suppose we are given a sequence of weighted undirected graphs with weight matrices $\mathbf{W}_1, \ldots, \mathbf{W}_T$ that, without loss of generality, share the same set of vertices. Let $f_i(A) = \phi_{\mathbf{W}_i}(A)$ be the the corresponding symmetric cut functions as defined in Equation 6.4.1. Note that by Equation 6.4.2, knowing f_i uniquely determines $\mathbf{W}_i$. (To handle the case of directed graphs, we can use a correspondence with undirected bipartite graphs of twice the size.)



(a) *Graph sketching*



(b) *Approximate optimization*

Figure 6.1: Experimental results. (a) Graph sketching of transitions of the Autonomous Systems graph. We plot number of random cuts observed vs. reconstruction error (Combined Type I + Type II error). During different transitions, the number Δ of changing edges varies. Notice how approximately 8Δ random observations suffice for perfect reconstruction. (b) Approximate submodular maximization in environmental monitoring. We wish to choose sets of locations with maximum mutual information. We compare the greedy algorithm optimizing the true functions, Fourier-sparse reconstructions obtained from n, $2n$, $4n$ and $8n$ samples with random selection. Notice that $8n$ samples already provide performance very close to the true objective.

As we have observed in Section 6.5, cut functions are contained in $\mathcal{H}_2$, and there is one edge for each nonzero Fourier coefficient. We can thus use Corollary 6.1 to reconstruct the graph by observing $\mathcal{O}(|k^{(t)}| \log^4 n)$ values of random cuts, where $k^{(t)}$ is the number of edges at time t. Note that while in practice, typically $k = \Omega(n)$, and for large graphs, we would require a proportionally large number of observations. If, however, we are interested in how a graph *changes* over time, and this change happens slowly, we use the fact that the difference $\mathbf{W}^{(t)} - \mathbf{W}^{(t+1)}$ is sparse.

In our experiments, we take a sequence of five snapshots of the Autonomous Systems graph [1]. Our experiments are performed on the subgraph induced by the 128 nodes with largest degree. We first pick an increasing number of sets at random. We then sketch the graphs at different time steps by computing the cut values associated with those sets. Since the cut function is linear in the edge weights, the difference in cut values corresponds to the cuts in the symmetric graph differences. We can therefore reconstruct the difference in the edge sets by using the reconstruction algorithm described in Section 6.5. Note that the number of changing edges varies from 62 to 245. Figure 6.1a presents the reconstruction error (in terms of the fraction of edges correctly classified as changing or not changing). For all transitions, exact recovery is possible, using a number of samples that is approximately a factor of 8 larger than the number of changing edges. Also, we observe that, consistently, with results in compressive sensing, a sharp phase transition occurs between a regime in which the error is close to 100%, and the regime in which perfect reconstruction occurs.

6.6.2 Approximate submodular optimization

Suppose a submodular function to be optimized is extremely expensive to evaluate, but can be approximated with our recontruction methods from random samples. Then one can evaluate the function on random samples to construct an approximation, and optimize the approximation. We test this approach on submodular function maximization in an environmental monitoring application. We consider the problem of selecting a small number of most informative observations for the purpose of spatial prediction. We take temperature data from the NIMS sensor node [HAG+06] deployed at a lake near the University of California, Merced. The environment is discretized in a set E of $n = 86$ locations. We train a nonstationary Gaussian Process using data from a single scan of the lake by the NIMS sensor node, using a

[1]downloaded from http://snap.stanford.edu

method described by [KSG08]. In order to quantify the informativeness of a set of locations $A \in 2^E$, we use the mutual information

$$f(A) = I(X_A; X_{E\setminus A}) = H(X_{E\setminus A}) - H(X_{E\setminus A} \mid X_A)$$

that quantifies the reduction of uncertainty in the unobserved locations $E \setminus A$ by taking into account the observations X_A at the selected observations. As shown by [KSG08], f is submodular and approximately monotonic (for small sets A). Therefore, an efficient greedy algorithm produces a set A_G with near-maximal informativeness. The algorithm proceeds by adding observations that maximally increase $f(A)$ until k observations have been selected [NWF78].

Unfortunately, computing mutual information $f(A)$ for the case of Gaussian processes requires solving a linear system of n variables, which is very expensive for large n. We consider approximating f by a low-order function. We evaluate f on an increasing number of sets, chosen uniformly at random, and then use the algorithm described in Section 6.5 to approximate $f \in \mathcal{H}_2$. Notice that even though f not exactly sparse, it appears to be well-approximated by a order 2 function: the best order 2 approximation explains approximately 86 % of its variance. In order to determine how well suited the approximate function is for optimization, we run the greedy algorithm on the approximation, and compare the resulting sets with the (provably near-optimal) solutions obtained by running the greedy algorithm on the original (expensive to evaluate) function f. As baseline, we also compare against the performance of sets chosen uniformly at random. Figure 6.1b presents the results of the experiment, using approximations obtained from n, $2n$, $4n$ and $8n$ random function evaluations. Notice that n and $2n$ function evaluations, not surprisingly, lead to poor performance. However, even $4n$ samples lead to strong performance, and $8n$ samples leads to solutions almost as good as those obtained when working with the true objective. These results indicate that the proposed approximations can perform very well even though the assumption of exact sparsity in the Fourier domain is not met.

6.6.3 Synthetic Submodular Recovery

We claimed in Section 6.4 that if a function is known to be submodular, then incorporating convex constraints implied by submodularity can improve the recovery of a function. We now describe some experiments on synthetic functions that demonstrate this claim empirically. We attempt the recovery a 3rd order submodular function by incorporating the constraints from Equation 6.4.6 into a convex recovery algorithm.

We take $n = 16$ and restrict to $\mathcal{H}_3$, therefore $p = 697$. By Proposition 5, we can check submodularity with 120 constraints. These numbers are small enough so that we can use a standard interior point method solver to get accurate solutions. We construct f by taking a function with i.i.d. Gaussian entries and then projecting it onto the cone $\mathcal{H}_2^- \cap \mathcal{H}_3$ to get a target synthetic function. The resulting projection is not exactly sparse; on average it has 200 ± 10 (out of 697 possible) nonzero Fourier coefficients. However, it is compressible, and so we can expect a small error even without recovering the support exactly. Then, given random function samples, we reconstruct the target function by minimizing the Fourier ℓ_1 norm, but vary what sets of additional constraints we apply. First, we simply solve Equation 6.3.4 with no additional constraints. For our second way, we assume oracle access to the signs of the Fourier coefficients and we enforce the known signs of the coefficients. Lastly, we enforce the constraints of Equation 6.4.6. The results are plotted in Figure 2. Enforcing submodularity significantly improves the recovery. It gives a relative error of less than 10^{-3} with only 300 measurements, and it recovers the support exactly with about 350. Using the ℓ_1 norm alone requires about 450 measurements just to get a relative error or 10^{-3}. The method with oracle access to the signs of the coefficients has better performance than standard ℓ_1, but is still not as good as the submodularity-enforcing method.

6.7 Related Work

Fourier analysis on the Boolean cube The problem of learning Boolean and pseudo-boolean functions has a long history with many special cases that have been studied, and the use of discrete Fourier analysis dates back to the work of [Man94] and [LMN93].

The specific problem of reconstructing graphs from few observations has received attention due to important applications in bioinformatics. [AC04]. The literature distinguishes *additive queries* (computing weight of all edges in a subgraph), and less powerful *cross-additive* queries (computing the weight of edges between two sets of vertices). Cuts are a special case of the latter. The literature also distinguishes *adaptive queries* (that can choose observations based on past observations) and less powerful *nonadaptive queries* (that have to commit to all

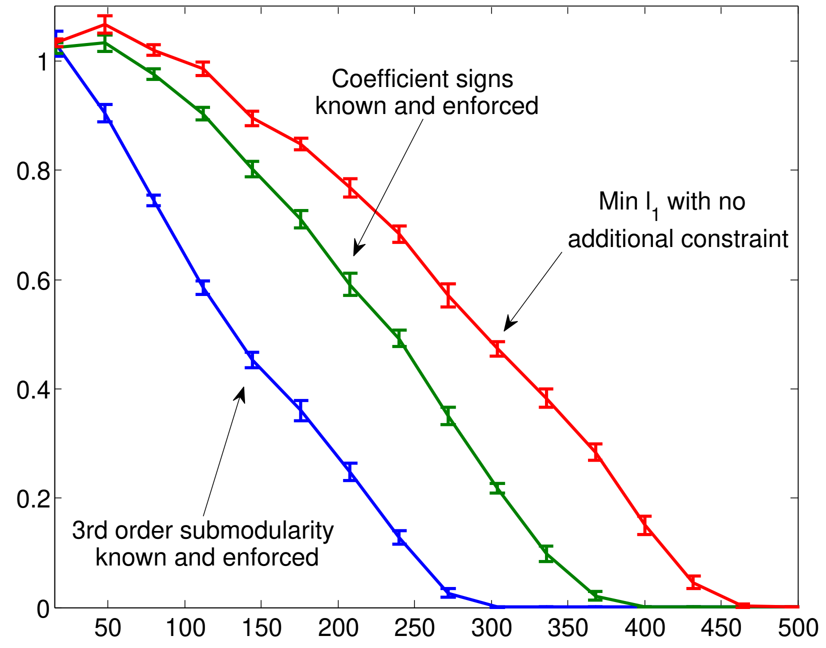


Figure 6.2: Empirical study of submodular constraints. Synthetic functions on a base set of 16 elements. Attempted recovery with differing types of constraints. The experiments were repeated with different random synthetic functions and different random measurments, and the mean relative error $\|f-g\|_2/\|f\|_2$ is plotted vs. number of random measurements.

observations in advance). In general, non-adaptive algorithms only requiring cross-additive queries are preferred (as these make the fewest assumptions, can be parallelized, etc.). For graphs with n nodes and k edges, an information theoretic lower bound of $\Omega\big(\frac{k\log(n^2/k)}{\log k}\big)$ additive (possibly adaptive) queries is known. [Maz10] provides an adaptive polynomial time algorithm that attains this optimal complexity in $\log n$ nonadaptive rounds. To our knowledge, the only existing *non-adaptive* algorithms with linear dependence on k are non-constructive (i.e., not polynomial time) [BM10]. This approach also requires *additive* queries. To our knowledge, ours is the first efficient nonadaptive approach (and furthermore only requires cross-additive queries).

Learning of pseudo-boolean functions (and associated hypergraphs) has been considered in [CJK11], which provides an almost tight *adaptive* algorithm for computing the Fourier coefficients of k-bounded pseudoboolean functions. [BM10] provide a non-adaptive, but also non-constructive approach, requiring additive queries.

Learning submodular functions Unfortunately, even without noise, there are *strong lower bounds*, limiting our expectations on learning *general* submodular functions. Without access to a data set of exponential size, it is not possible to approximate general submodular functions to a factor better than $\Omega(\sqrt{n}/\log n)$ [GHIM09]. On a more positive side, if the function is Lipschitz, and sets are sampled uniformly at random, then for any $\varepsilon > 0$, a $\mathcal{O}(\log \frac{1}{\varepsilon})$ approximation can be achieved on a fraction of at least $1-\varepsilon$ of all sets [BH11]. However, for optimization purposes, a guarantee that the approximation is of high quality on only a subset (even a large subset) of sets is problematic, since typically nothing can be inferred about the resulting minimizer. The problem of approximating a general submodular function by a simpler one for the purpose of efficient minimization is studied in [JLB11], who do not exploit the special structure of Fourier-sparse functions.

Compressive sensing There has been vast interest in sparse reconstruction and compressive sensing [Don06, CW08]. But traditionally this has been motivated by sparsity of signals as a trigonometric polynomial or in the wavelet domain. However, we are unaware of any work directly applying these ideas to discrete cube. If work on sublinear Fourier transforms as of [GGI+02] can be thought of as applying the ideas from learning sparse boolean functions to sparse trigonmetric polynomials, then our work can be thought of as doing the reverse. Opening up a toolbox of new methods for this domain is the main contribution of this chapter.

6.8 Conclusion

We have considered the problem of reconstructing set functions with decaying Fourier (Hadamard-Walsh) spectrum, from a small number of possibly noisy observations. By leveraging recent results from random matrices and sparse reconstruction, we have shown that standard algorithms can be used to obtain perfect reconstruction, with a number of samples that scales linearly with the support size of the Fourier spectrum. This insight allows us to open up a vast toolbox of modern optimization methods for learning set functions, which previously has been mostly the domain of purely theoretical investigation. For example, our results imply that standard ℓ_1 minimization can be used to reconstruct a sparse graph from observing the values of a number of random cuts, which (up to logarithmic factors) matches information-theoretic lower bounds in [Maz10]. Furthermore, we show that other properties, such as submodularity and symmetry, imply structure among the Fourier coefficients that can be exploited to reduce sample complexity, as well as speed up reconstruction algorithms.

We demonstrate the effectiveness of our approach on two applications, showing that we can indeed sketch changes in real-world networks by measuring random cuts, and that we can obtain useful approximations of expensive-to-compute set functions for the purpose of optimization.

Chapter 7

Conclusion

One of our main objectives in this work has been to bridge the gap between the two extremes of submodular functions: on one hand there are very basic or specialized functions that admit simple and practical minimization algorithms, but are fairly limited in what they can describe, and, on the other, there are the those for which no specialized minimization algorithms are known. For the latter case, the only algorithms that have polynomial upper bounds on the running time are impractical for all but very small problems, and the methods which seem to work have no guarantees. Our intent has been to the extend the classes of submodular functions for which there exist fairly efficient algorithms, and to that end in Chapter 4 and 5 we developed several novel techniques for submodular minimization using convex analysis.

The SLG algorithm of Chapter 4 gives a method for minimizing sums of functions of the form $\phi(\mathbf{w}(A))$: these are compositions of nonnegative modular functions $\mathbf{w}$ with concave functions ϕ. Admittedly, such functions are always representable as a minimum of a second order submodular function, so in principle they can be solved via graph cut algorithms. However, if the concave function ϕ is not piecewise linear, in order to represent the function exactly, one needs to introduce a breakpoint (corresponding to an extra vertex in the graph) for every possible value of $\mathbf{w}(A)$, which, in the worst case, is 2^n. The main innovation of our algorithm is that it remains efficient and independent of the number of breakpoints. We do this by computing a gradient of a smoothed version of the Lovász extension and applying an accelerated descent method.

In Chapter 5, our main result demonstrates that there is a great deal of flexibility in choosing a barrier function to minimize a submodular function via convex programming. It might be useful, in fact, to adapt the barrier function to the submodular function. In analogy to matrix preconditioning: would this help address scaling issues, and how could we estimate an appropriate choice for the barrier function?

The major question we attempted to address, and one which remains unresolved, is how can we try to discover better representations of submodular functions? A common theme of this thesis is the utility of having a simple description of the base polytope of a submodular function. We expect such descriptions to be useful both for learning a submodular function and projecting onto the base polytope (essentially the same problem as submodular minimization).

Most of the functions in our experiments are variations of graph representable functions such as in Equation 3.3.13. Our interest has not been in trying to extend graph cut algorithms but to take advantage of the efficient base polytope representation of Equation 3.3.14. By efficient, we mean refer to the fact that the number of variables and additional constraints scales linearly with the number of nonzero coefficients in the specification. Can we find any other classes of submodular functions that also have efficient lifted representations—possibly in terms of the second order cone or the semidefinite cone? And if such classes exist, do the any of results in Chapter 6 generalize to allow one to learn such a representation in a reasonable time with a reasonable number of observations?

Lastly, we would like to pose one core question to that seems simple but remains open to our knowledge—what is the smallest family of submodular functions such that *every* submodular function is in their conic hull? Equivalently, what is the smallest number of half-spaces that intersect to form the submodular polar cone $\left(\mathcal{H}_2^-\right)^\circ$? We know this family must exist, and it must be finite, albeit exponential in size.

Appendix A

Matroid Theory

Matroids are a class of set systems which are intimately connected to submodular set functions. The name "matroid" stems from the fact that they can be treated as an abstract generalization of matrices. However, there are several different equivalent definitions of matroids, some of which are quite unexpected. For a gentle introduction to the subject, see [GM12]. Since our focus is on submodularity, we give merely the original definition of matroid introduced by Whitney [Whi35] (in terms of independent sets), as well as a formulation (in terms of the existence of a rank function) which directly invokes submodularity.

A matroid $M = (E, \mathcal{I})$ is defined by a ground set E together with a collection of subsets $\mathcal{I} \subseteq 2^E$ that satisfies the following properties:

Properties A.1. Matroid Independence Axioms

I-1. $\emptyset \in \mathcal{I}$

I-2. $A \subset B$ and $B \in \mathcal{I} \;\Rightarrow\; A \in \mathcal{I}$

I-3. $A, B \in \mathcal{I}$ with $|A| < |B| \;\Rightarrow\; \exists\, b \in B \setminus A$ such that $A + b \in \mathcal{I}$

It is an exercise in elementary linear algebra to show that if E is a set of vectors from a common vector space, then the collection of independent subsets of vectors satisfies the Properties A.1. For general matroids, the sets in the collection $\mathcal{I}$ are called the independent sets of a matroid. If a set of vectors are expressed in the form of a matrix, and the independent sets of those vectors are in one-to-one correspondence with the independent sets of a matroid, then that matrix is said to represent the matroid. Matroids are strictly more general than matrices in that not every matroid can be represented as a matrix, though the simplest example requires $|E| \geq 8$.

An alternative characterization of a matroid is through the properties of an associated set function. One can show that a set E is the ground set for a matroid if and only if it admits a set function $r \in \mathcal{H}$ with the following properties:

Properties A.2. Matroid Rank Axioms

R-1. $r(\emptyset) = 0$.

R-2. $r(A+b) - r(A) \in \{0,1\}$ for all $A \in 2^E$, $b \in E \setminus A$.

R-3. $r(A+b+c) - r(A+b) - r(A+c) + r(A) \leq 0$ for all $A \in 2^E$, $b, c \in E \setminus A$.

Of particular note is the third property: submodularity.

As an example, if E is a set of vectors, then the Properties A.2 are satisfied by the set function equal to the dimension of the subspace spanned by vectors in a set (a.k.a. the rank). For general matroids, the rank of a set is defined as the size of the largest independent subset:

$$r(A) = \max_{\substack{B \in \mathcal{I} \\ B \subseteq A}} |B| \tag{A.0.1}$$

One can check that for any matroid, the rank function defined this way gives rise to a set function satisfying Properties A.2.

Conversely, given a rank function satisfying Properties A.2, one can define independent sets as those whose size equals their rank:

$$\mathcal{I} = \{A \in 2^E \mid r(A) = |A|\} \tag{A.0.2}$$

Then one can check that this collection forms the independence system of a matroid; that is, Properties A.1 are satisfied.

Other defining features of matroids are their circuits (minimal dependent sets) and their bases (maximal independent sets). The latter is significant for the theory of submodular functions, since the set of bases of a matroid forms corresponds to the vertices of the base polytope (as defined in Equation 3.3.2) of the rank function.

One extremely important generalization of matroids are polymatroids, as introduced by Edmonds [Edm70]. If the independence system of a matroid is considered to be a subset of the Boolean cube $\{0,1\}^n$, then a polymatroid is an analogous subset of $\mathbb{Z}_+^n$ or $\mathbb{R}_+^n$. In the case of real vectors, it is equivalent to the base polytope of a nonnegative nondecreasing submodular function.

Bibliography

[ABK$^+$04] N. Alon, R. Beigel, S. Kasif, S. Rudich, and B. Sudakov, *Learning a hidden matching*, SIAM J. Comput. **33** (2004), no. 2, 487–501. 6.4.2

[AC04] D. Angluin and J. Chen, *Learning a hidden graph using o (log n) queries per edge*, Learning Theory (2004), 210–223. 6.7

[AT03] A. Auslender and M. Teboulle, *Asymptotic cones and functions in optimization and variational inequalities*, Springer Monographs in Mathematics, Springer-Verlag, New York, 2003. 2.1

[AT06] ______, *Interior gradient and proximal methods for convex and conic optimization*, SIAM Journal on Optimization **16** (2006), no. 3, 697–725. 6.5

[Bac10] F. Bach, *Structured sparsity-inducing norms through submodular functions*, Advances in Neural Information Processing Systems 23 (J. Lafferty, C. K. I. Williams, J. Shawe-Taylor, R.S. Zemel, and A. Culotta, eds.), 2010, pp. 118–126. 4.2

[Bac11] ______, *Learning with submodular functions: A convex optimization perspective*, CoRR **abs/1111.6453** (2011). 3.3.1

[BBC11] S. Becker, J. Bobin, and E.J. Candès, *Nesta: A fast and accurate first-order method for sparse recovery*, SIAM J. Img. Sci. **4** (2011), no. 1, 1–39. 4.5.2

[BCG11] S. Becker, E. J. Candès, and M. Grant, *Templates for convex cone problems with applications to sparse signal recovery*, Mathematical Programming Computation **3** (2011), 165–218. 6.5

[BH11] M.F. Balcan and N.J.A. Harvey, *Learning submodular functions*, Proceedings of the 43rd annual ACM symposium on Theory of computing, ACM, 2011, pp. 793–802. 6.1, 6.7

[BM10] N.H. Bshouty and H. Mazzawi, *Optimal query complexity for reconstructing hypergraphs*, STACS 2010: 27th International Symposium on Theoretical Aspects of Computer Science, LIPIcs., vol. 5, Leibniz-Zent. Inform., 2010, pp. 143–154. 6.4.2, 6.7

[Brè67] L. M. Brègman, *A relaxation method of finding a common point of convex sets and its application to the solution of problems in convex programming*, Ž. Vyčisl. Mat. i Mat. Fiz. **7** (1967), 620–631. 2.2

[CJK11] S.S. Choi, K. Jung, and J.H. Kim, *Almost tight upper bound for finding fourier coefficients of bounded pseudo-boolean functions*, Journal of Computer and System Sciences **77** (2011), no. 6, 1039–1053. 6.3, 6.7

[CK10] S.S. Choi and J.H. Kim, *Optimal query complexity bounds for finding graphs*, Artificial Intelligence **174** (2010), no. 9, 551–569. 6.4.2

[CKKL12] M. Cheraghchi, A. Klivans, P. Kothari, and H.K. Lee, *Submodular functions are noise stable*, Proceedings of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms, SODA '12, SIAM, 2012, pp. 1586–1592. 6.4.3

[CN07] F. A. Chudak and K. Nagano, *Efficient solutions to relaxations of combinatorial problems with submodular penalties via the lovász extension and non-smooth convex optimization*, SODA, 2007, pp. 79–88. 4.2

[CP11] P.L. Combettes and J.C. Pesquet, *Proximal splitting methods in signal processing*, Fixed-Point Algorithms for Inverse Problems in Science and Engineering (2011), 185–212. 2.2

[CRT06] E.J. Candès, J. Romberg, and T. Tao, *Robust uncertainty principles: exact signal reconstruction from highly incomplete frequency information*, Information Theory, IEEE Transactions on **52** (2006), no. 2, 489–509. 6.3

[CT05] E.J. Candès and T. Tao, *Decoding by linear programming*, Information Theory, IEEE Transactions on **51** (2005), no. 12, 4203–4215. 6.3

[CW08] E.J. Candès and M.B. Wakin, *An introduction to compressive sampling*, Signal Processing Magazine, IEEE **25** (2008), no. 2, 21–30. 6.7

[Don06] D.L. Donoho, *Compressed sensing*, Information Theory, IEEE Transactions on **52** (2006), no. 4, 1289–1306. 6.3, 6.7

[DSSSC08] J. Duchi, S. Shalev-Shwartz, Y. Singer, and T. Chandra, *Efficient projections onto the l1-ball for learning in high dimensions*, Proceedings of the 25th international conference on Machine learning, ICML '08, 2008, pp. 272–279. 5.4.2

[Edm70] J. Edmonds, *Submodular functions, matroids, and certain polyhedra*, Combinatorial Structures and their Applications (Proc. Calgary Internat. Conf., Calgary, Alta., 1969), Gordon and Breach, New York, 1970, pp. 69–87. 1, 3.3.1, 3.3.2, A

[EVGW$^+$] M. Everingham, L. Van Gool, C.K.I. Williams, J. Winn, and A. Zisserman, *The PASCAL Visual Object Classes Challenge 2009 (VOC2009) Results*, http://www.pascal-network.org/challenges/VOC/voc2009/workshop/index.html. 4.7

[FD05] D. Freedman and P. Drineas, *Energy minimization via graph cuts: settling what is possible*, Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on, vol. 2, june 2005, pp. 939 – 946 vol. 2. 4.1

[FH05] S. Foldes and P.L. Hammer, *Submodularity, Supermodularity, and Higher-Order Monotonicities of Pseudo-Boolean Functions*, Mathematics of Operations Research **30** (2005), no. 2, 453. 3.2.2

[FI03] L. Fleischer and S. Iwata, *A push-relabel framework for submodular function minimization and applications to parametric optimization*, Discrete Appl. Math. **131** (2003), no. 2, 311–322, Submodularity. 4.7

[FI11] S. Fujishige and S. Isotani, *A submodular function minimization algorithm based on the minimum-norm base*, Pac. J. Optim. **7** (2011), no. 1, 3–17. 3.4.2, 4.7

[Fis67] P.C. Fishburn, *Interdependence and additivity in multivariate, unidimensional expected utility theory*, International Economic Review **8** (1967), no. 3, 335–342. 6.1

[Fou10] S. Foucart, *A note on guaranteed sparse recovery via âĎŞ1-minimization*, Applied and Computational Harmonic Analysis **29** (2010), no. 1, 97–103. 6.3

[Fuj05] S. Fujishige, *Submodular functions and optimization*, second ed., Annals of Discrete Mathematics, vol. 58, Elsevier B. V., Amsterdam, 2005. MR 2171629 (2006d:90098) 3.3, 3.3.1

[GGI$^+$02] A.C. Gilbert, S. Guha, P. Indyk, S. Muthukrishnan, and M. Strauss, *Near-optimal sparse fourier representations via sampling*, Proc. STOC, 2002. 6.7

[GHIM09] M.X. Goemans, N.J.A. Harvey, S. Iwata, and V. Mirrokni, *Approximating submodular functions everywhere*, Proceedings of the twentieth Annual ACM-SIAM Symposium on Discrete Algorithms, Society for Industrial and Applied Mathematics, 2009, pp. 535–544. 6.1, 6.7

[GK00] V. Grebinski and G. Kucherov, *Optimal Reconstruction of Graphs under the Additive Model.*, Algorithmica **28** (2000), no. 1, 104–124. 6.4.2

[GLS81] M. Grötschel, L. Lovász, and A. Schrijver, *The ellipsoid method and its consequences in combinatorial optimization*, Combinatorica **1** (1981), no. 2, 169–197. 1, 3.3.2, 3.4.1

[GM12] G. Gordon and J. McNulty, *Matroids: a geometric introduction*, Cambridge University Press, Cambridge, 2012. A

[GMR00] M. Grabisch, J.L. Marichal, and M. Roubens, *Equivalent representations of set functions*, Mathematics of Operations Research **25** (2000), no. 2, 157–178. 3.2.3, 3.2.4

[Hač79] L.G. Hačijan, *A polynomial algorithm in linear programming*, Dokl. Akad. Nauk SSSR **244** (1979), no. 5, 1093–1096. 3.4.1

[HAG$^+$06] T.C. Harmon, R.F. Ambrose, R.M. Gilbert, J.C. Fisher, M. Stealey, and W.J. Kaiser, *High Resolution River Hydraulic and Water Quality Characterization using Rapidly Deployable Networked Infomechanical Systems (NIMS RD)*, Tech. Report 60, CENS, 2006. 6.6.2

[HH92] P.L. Hammer and R. Holzman, *Approximations of pseudo-Boolean functions; applications to game theory*, ZOR Zeitschrift für Operations Research Methods and Models of Operations Research **36** (1992), no. 1, 3–21. 3.2.3

[HUL93] Jean-Baptiste Hiriart-Urruty and Claude Lemaréchal, *Convex analysis and minimization algorithms. II*, Grundlehren der Mathematischen Wissenschaften [Fundamental Principles of Mathematical Sciences], vol. 306, Springer-Verlag, Berlin, 1993, Advanced theory and bundle methods. 2.1

[IFF01] S. Iwata, L. Fleischer, and S. Fujishige, *A combinatorial strongly polynomial algorithm for minimizing submodular functions*, J. ACM **48** (2001), no. 4, 761–777. 1, 3.4.1

[II07] T. Itoko and S. Iwata, *Computational geometric approach to submodular function minimization for multiclass queueing systems*, Integer Programming and Combinatorial Optimization (Matteo Fischetti and DavidP. Williamson, eds.), Lecture Notes in Computer Science, vol. 4513, Springer Berlin Heidelberg, 2007, pp. 267–279. 4.3, 4.4.3

[IO09] S. Iwata and J. B. Orlin, *A simple combinatorial algorithm for submodular function minimization*, Proceedings of the twentieth Annual ACM-SIAM Symposium on Discrete Algorithms (Philadelphia, PA, USA), SODA '09, Society for Industrial and Applied Mathematics, 2009, pp. 1230–1237. 1, 3.4.1

[JLB11] Stefanie Jegelka, Hui Lin, and Jeff A. Bilmes, *On fast approximate submodular minimization*, Advances in Neural Information Processing Systems 24 (J. Shawe-Taylor, R.S. Zemel, P. Bartlett, F.C.N. Pereira, and K.Q. Weinberger, eds.), 2011, pp. 460–468. 3.4.3, 6.7

[JM93] D.S. Johnson and C.C. McGeoch (eds.), *Network flows and matching: First dimacs implementation challenge*, American Mathematical Society, Boston, MA, USA, 1993. 4.1, 4.7

[KK80] A.K. Kelmans and B.N. Kimelfeld, *Multiplicative submodularity of a matrix's principal minor as a function of the set of its rows and some combinatorial applications*, Discrete Mathematics **44** (1980), no. 1, 113–116. 6.1

[KKT07] P. Kohli, M.P. Kumar, and P.H.S. Torr, *P3 beyond: Solving energies with higher order cliques*, Computer Vision and Pattern Recognition, 2007. CVPR '07. IEEE Conference on, june 2007, pp. 1 –8. 4.3

[KLT09] P. Kohli, L. Ladický, and P.H. Torr, *Robust higher order potentials for enforcing label consistency*, Int. J. Comput. Vision **82** (2009), no. 3, 302–324. 4.3

[Kra10] A. Krause, *Sfo: A toolbox for submodular function optimization*, J. Mach. Learn. Res. **11** (2010), 1141–1144. 4.7

[KSG08] A. Krause, A. Singh, and C. Guestrin, *Near-Optimal Sensor Placements in Gaussian Processes: Theory, Efficient Algorithms and Empirical Studies*, Journal of Machine Learning Research **9** (2008), 235–284. 6.1, 6.6.2

[KZ04] V. Kolmogorov and R. Zabin, *What energy functions can be minimized via graph cuts?*, Pattern Analysis and Machine Intelligence, IEEE Transactions on **26** (2004), no. 2, 147–159. 3.4.3

[LMN93] N. Linial, Y. Mansour, and N. Nisan, *Constant depth circuits, Fourier transform, and learnability*, Journal of the ACM (JACM) **40** (1993), no. 3, 607–620. 6.7

[Lov83] L. Lovász, *Submodular functions and convexity*, Mathematical programming: the state of the art, Springer, Berlin, 1983, pp. 235–257. 1, 3.3.2, 4.2, 4.2, 4.2, 6.1, 6.4.3

[Man94] Y. Mansour, *Learning boolean functions via the fourier transform*, Theoretical advances in neural computation and learning (1994), 391–424. 6.7

[Maz10] H. Mazzawi, *Optimally reconstructing weighted graphs using queries*, Proc. SODA, 2010. 6.7, 6.8

[Mur03] K. Murota, *Discrete convex analysis*, SIAM Monographs on Discrete Mathematics and Applications, Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 2003. 1

[Nes05] Y. Nesterov, *Smooth minimization of non-smooth functions*, Mathematical Programming **103** (2005), 127–152. 4.1, 4.2, 4.5.2, 4.5.2

[NWF78] G.L. Nemhauser, L.A. Wolsey, and M.L. Fisher, *An analysis of approximations for maximizing submodular set functions - I*, Mathematical Programming **14** (1978), no. 1, 265–294. 3.3, 6.6.2

[PK90] P.M. Pardalos and N. Kovoor, *An algorithm for a singly constrained class of quadratic programs subject to upper and lower bounds*, Mathematical Programming **46** (1990), 321–328. 5.4.2

[Que98] M. Queyranne, *Minimizing symmetric submodular functions*, Mathematical Programming **82** (1998), 3–12. 3.4.3, 4.1

[Rau10] H. Rauhut, *Compressive sensing and structured random matrices*, Theoretical foundations and numerical methods for sparse recovery, Radon Ser. Comput. Appl. Math., vol. 9, Walter de Gruyter, Berlin, 2010, pp. 1–92. 6.3

[Roc70] R.T. Rockafellar, *Convex analysis*, Princeton Mathematical Series, No. 28, Princeton University Press, Princeton, N.J., 1970. 2.1

[RV08] M. Rudelson and R. Vershynin, *On sparse reconstruction from Fourier and Gaussian measurements*, Communications on Pure and Applied Mathematics **61** (2008), no. 8, 1025–1045. 6.3

[Sch03] A. Schrijver, *Combinatorial optimization. Polyhedra and efficiency. Vol. B*, Algorithms and Combinatorics, vol. 24, Springer-Verlag, Berlin, 2003, Matroids, trees, stable sets, Chapters 39–69. 3.3.1, 6.1

[SK10] P. Stobbe and A. Krause, *Efficient minimization of decomposable submodular functions*, NIPS, 2010, pp. 2208–2216. 1.1, 1.2

[SK12] ———, *Learning fourier sparse set functions*, Journal of Machine Learning Research - Proceedings Track **22** (2012), 1125–1133. 1.2

[SWRC09] J. Shotton, J. Winn, C. Rother, and A. Criminisi, *Textonboost for image understanding: Multi-class object recognition and segmentation by jointly modeling texture, layout, and context*, International Journal of Computer Vision **81** (2009), 2–23. 4.3, 4.7

[Tro04] J.A. Tropp, *Greed is good: algorithmic results for sparse approximation.*, IEEE Transactions on Information Theory **50** (2004), no. 10, 2231–2242. 6.5

[TW10] J.A. Tropp and S.J. Wright, *Computational Methods for Sparse Solution of Linear Inverse Problems*, Proceedings of the IEEE **98** (2010), no. 6, 948–958. 6.5

[Ver12] R. Vershynin, *Introduction to the non-asymptotic analysis of random matrices*, Compressed sensing, Cambridge Univ. Press, Cambridge, 2012, pp. 210–268. 6.1, 6.3

[Whi35] H. Whitney, *On the Abstract Properties of Linear Dependence*, Amer. J. Math. **57** (1935), no. 3, 509–533. A