Convex Modeling with Priors

by

Benjamin Recht

B.S., University of Chicago (2000)
M.S., Massachusetts Institute of Technology (2002)

Submitted to the Media Arts and Sciences
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2006

Author .....
Media Arts and Sciences
April 14, 2006

Certified by .....
Neil Gershenfeld
Associate Professor
Thesis Supervisor

Accepted by .....
Andrew B. Lippman
Chairman, Departmental Committee on Graduate Students

Convex Modeling with Priors

by

Benjamin Recht

Submitted to the Media Arts and Sciences
on April 14, 2006, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

Abstract

As the study of complex interconnected networks becomes widespread across disciplines, modeling the large-scale behavior of these systems becomes both increasingly important and increasingly difficult. In particular, it is of tantamount importance to utilize available prior information about the system's structure when building data-driven models of complex behavior. This thesis provides a framework for building models that incorporate domain specific knowledge and glean information from unlabelled data points.

I present a methodology to augment standard methods in statistical regression with priors. These priors might include how the output series should behave or the specifics of the functional form relating inputs to outputs. My approach is optimization driven: by formulating a concise set of goals and constraints, approximate models may be systematically derived. The resulting approximations are convex and thus have only global minima and can be solved efficiently. The functional relationships amongst data are given as sums of nonlinear kernels that are expressive enough to approximate any mapping. Depending on the specifics of the prior, different estimation algorithms can be derived, and relationships between various types of data can be discovered using surprisingly few examples.

The utility of this approach is demonstrated through three exemplary embodiments. When the output is constrained to be discrete, a powerful set of algorithms for semi-supervised classification and segmentation result. When the output is constrained to follow Markovian dynamics, techniques for nonlinear dimensionality reduction and system identification are derived. Finally, when the output is constrained to be zero on a given set and non-zero everywhere else, a new algorithm for learning latent constraints in high-dimensional data is recovered.

I apply the algorithms derived from this framework to a varied set of domains. The dissertation provides a new interpretation of the so-called Spectral Clustering algorithms for data segmentation and suggests how they may be improved. I demonstrate the tasks of tracking RFID tags from signal strength measurements, recovering the pose of rigid objects, deformable bodies, and articulated bodies from video sequences. Lastly, I discuss empirical methods to detect conserved quantities and learn constraints defining data sets.

Thesis Supervisor: Neil Gershenfeld
Title: Associate Professor

Convex Modeling with Priors

by

Benjamin Recht

PhD Thesis

Signature Page

Thesis Advisor .....

Neil Gershenfeld

Associate Professor

Department of Media Arts and Sciences

Massachusetts Institute of Technology

Thesis Reader .....

John Doyle

John G. Braun Professor

Departments of Control and Dynamical Systems, Electrical Engineering, and BioEngineering

California Institute of Technology

Thesis Reader .....

Pablo Parrilo

Associate Professor

Department of Electrical Engineering and Computer Science

Massachusetts Institute of Technology

Acknowledgments

I would first like to thank the members of my committee for providing invaluable guidance and support. My advisor, Neil Gershenfeld, encouraged me to pursue my diverse interests and provided a stimulating environment in which to do so. John Doyle took me under his wing and introduced me to the wide world of robustness. Pablo Parrilo shared his many insights and creative suggestions on this document and on the score of papers we still have left in the queue.

The work in this thesis arose out of a long-running collaboration with Ali Rahimi. Most of the results contained herein are distilled from papers we have written together or ideas composed in late nights of brainstorming and coding (with some bickering). I'd like to thank him for such a fruitful collaboration. This document benefited from a careful reading by Ryan Rifkin who provided many useful comments and corrections. Aram Harrow suggested a simple proof of Theorem 4.2.4. Along with being my co-conspirator on the Audiopad project, James Patten provided the hardware and his time for the acquisition of the Sensetable data in Chapter 5. Andy Sun and Jon Santiago assisted in the collection of the Resistofish data also discussed in Chapter 5. Kenneth Brown, Bill Butera, Constantine Caramanis, Waleed Farahat, Tad Hirsch, and Dan Paluska also provided helpful feedback.

I'd especially like to thank Raffaello D'Andrea for his friendship and mentorship during my graduate career. I'd like to thank everyone else I have collaborated with during my stay at MIT including Ethan Bordeaux, Isaac Chuang, Brian Chow, Chris Csikszentmihalyi, Trevor Darrell, Saul Griffith and Squid Labs, Hiroshii Ishii, Seth Lloyd, Cameron Marlow, Jim McBride, Ryan McKinley, Ravi Pappu, Jason Taylor, Noah Vawter, Brian Whitman, and the students of the MIT Media Lab. I have learned more in these collaborations with my peers than anywhere else in my graduate career. I would also like to acknowledge my other fellow travellers in the Physics and Media Group: Rich F., Ara K., Raffi K., Femi O., Rehmi P., Manu P., Matt R., Amy S., and Ben V.

None of this work would have been possible without the diligent staff of the CBA office. I'd like to thank Susan Murphy-Bottari, Kelly Maenpaa, and Mike Houlihan for all of their help, hard work, and support. I also extend my gratitude to Linda Peterson and Pat Solakoff in the MAS office for making it easier to leap over the hurdles that accompany the graduate school process.

My involvment in the Boston electronic music scene has served as an important counterpoint to and release from my academic work. Notable shout outs go to The Fun Years, The Dan Bensons Project, Mike Uzzi, The Saltmine, Unlockedgroove, The DSP Music Syndicate, The Appliance of Science, non-event, Beat Research, Spectrum, Jake Trussell, Anthony Flackett, Collision, Don Mennerich, Eric Gunther, Fred Giannelli, and Stewart Walker. We are the reason Boston is the drone capital of the universe.

Of course, I am deeply indebted to Mom and Dad for their endless support. I'd like to thank my sister Marissa for putting up with my bad attitude longer than any reasonable person would have. And finally, I'd like to thank Lauren, without whom I would have probably finished this thesis, but it wouldn't have been half as good.

This work was supported in part by the Center for Bits and Atoms (NSF CCR-0122419), ARDA/DTO (F30602-30-2-0090), and the MITRE Corporation (0705N7KZ-PB).

Contents

List of Figures

List of Tables

Chapter 1

Introduction

We are currently building systems that produce more data at higher rates than ever before. With the advent of faster computers and a deluge of measurements from sensors, surveys, and gene arrays, building simple models to describe complex physical phenomena is a daunting challenge. Deriving simple models from data with principled tools that leverage *a priori* knowledge rather than expert tuning and annotation is of tantamount importance. Gaining intuitive understanding of these models and the modeling tools is equally important.

In this thesis, I will argue that if I can pose a modeling problem in terms of structured goals and constraints, then I can apply tools from convex optimization to automatically generate algorithms to efficiently fit the best model to my data. In many regards, the main contribution is in problem posing. Not all problems can be posed as convex optimizations, but I will demonstrate through a variety of applications that this methodology is widely applicable and very powerful.

Recently, a great deal of interest has emerged around modeling complex systems with *mathematical programming* – the applied mathematics concerned with optimizing cost functions under a set of constraints. Many modeling, analysis, and design questions can be phrased as a series of goals and constraints. What is the shortest route from my house to work? What is the optimal strategy for managing congestion on the internet while maintaining user satisfaction [63]? How can an array of oscillators maximize their phase coherence [47]? Using the tools of mathematical

programming, a well-phrased problem statement alone can provide sufficient information to guarantee properties of system behavior, derive protocols for achieving optimal performance, and verify the convergence of the dynamics that solve the optimization.

A special class of mathematical programs are the *convex programs*. Notable convex programs are the well-known problems of least-squares and linear programming for which very efficient algorithms exist. Building on these two examples, algorithms for convex optimization have matured rapidly in the last couple of decades. Today, the solution of convex programs is typically no more complicated than that of matrix inversion and the techniques have been applied in fields as diverse as automatic control, electronic circuit design, economics, estimation, statistical machine learning, and network design [15]. This puts the burden on the applied mathematician to either phrase a problem in a convex form or to recognize when this is not possible in an efficient manner.

Of course, not all problems can be phrased as convex optimizations. There are well-known classes of problems believed to be intractable independent of the applied solution technique. However, convex techniques have produced extremely good approximations to many known hard problems. Such approximations, called *relaxations*, provide guaranteed error bounds to intractable problems. Beginning with the work of Goemans and Williamson on approximating the NP-HARD problems MAX-CUT and MAX-2-SAT [37] in combinatorial optimization, an industry of approximating intractable problems to high tolerance has developed [31, 50, 52]. Whereas heuristic searches like genetic algorithms and simulated annealing provided no insight into the values they would output, the convex methods produce guaranteed error margins.

Inspired by these relaxations, this thesis will develop tools that exploit convexity to build approximate data-driven models incorporating expert *a priori* knowledge. My approach is cost function driven. By summarizing the modeling problem as a set of goals and constraints, I will systematically produce a convex representation of the problem of tractable size and an algorithm in the new representation which approximates the original formulation.

1.1 Contributions and Organization

In Chapter 2, I will provide a brief review of the mathematical foundations upon which this thesis rests. Beginning with an overview of convexity, I will summarize the theory of convex relaxations and Lagrangian duality, and I will discuss the connections with function learning on *Reproducing Kernel Hilbert Spaces* (RKHS), a powerful functional representation where the optimal mappings are sums of functions centered around each data point.

In Chapter 3, I will present a powerful cost function that can be applied to a vast array of data-driven modeling problems and can be optimized in a principled way. By augmenting the simple problem of fitting the best function in an RKHS to a set of data with a set of priors, I will produce a very general and powerful cost function for modeling with priors. This optimization seeks to jointly find the best model relating data to attributes, the labels of the unlabelled data, and the best space of functions to represent the relationship. The optimization will be convex in all of these arguments. In particular, I will present several novel results about learning kernel functions. I will provide a general formulation of the learning algorithms that may be solved with semidefinite programming. I will derive solutions to learning the width of Gaussian kernels. Finally, I will show how to search for the best polynomial kernel using semidefinite programming.

In Chapter 4, the first prior on the output will be presented. By restricting the desired outputs to be binary labels, a family of optimization problems for segmentation, clustering, and transductive classification can be derived. I will show that even though the prior is so simple, all of the resulting optimizations are NP-HARD and no efficient algorithm to solve them exactly can be expected. In turn, I will present approximation algorithms using semidefinite programming. These semidefinite programs may be prohibitively slow for very large numbers of examples, so I will also present an additional family of relaxations that reduce to eigenvalue problems. These eigenvalue problems recover the well-known Spectral Clustering algorithms. This function learning interpretation provides insight into when and how such algorithms

fail as well as how they can be corrected.

In Chapter 5, I describe a dynamics prior that results in a family of semi-supervised regression algorithms that learn mappings between time series. These algorithms are applied to tracking, where a time series of observations from sensors is transformed to a time series describing the pose of a target. Instead of defining and implementing such transformations for each tracking task separately, the algorithms learn memoryless transformations of time series from a few example input-output mappings. The learning procedure is fast and lends itself to a solution by least-squares or by the solution of an eigenvalue problem. I discuss the relationships with nonlinear system identification and manifold learning techniques. The utility of the dynamics prior is demonstrated on the tasks of tracking RFID tags from signal strength measurements, recovering the pose of rigid objects, deformable bodies, and articulated bodies from video sequences. For such tasks, these new algorithms require significantly fewer examples compared to fully-supervised regression algorithms or semi-supervised learning algorithms that do not take the dynamics of the output time series into account.

Finally, **in Chapter 6**, I consider the problem of learning constraints satisfied by data. I show how to learn a space of functions that is constant on the data set and suggest how to select a maximal set of constraints. I show how this algorithm can learn descriptions of data sets that are not parsable by existing manifold learning algorithms. In particular, I show that this new algorithm can learn manifolds that are not diffeomorphic to Euclidean space.

1.2 Notation

This section serves as a glossary for the mathematical symbols used in the text. $\mathbb{R}$ denotes the real numbers and $\mathbb{Z}$ denotes the integers.

Vectors will be denoted by bold face lower case letters. Matrices will be denoted by bold face capital letters or capital Greek letters. The components of vectors and matrices will be denoted by subscripted non-bold letters. For example, the component

of the matrix $\mathbf{A}$ in the ith row and jth column is denoted A_{ij}.

The transpose of a matrix $\mathbf{A}$ will be denoted by $\mathbf{A}^\top$. The inverse of $\mathbf{A}$ is denoted $\mathbf{A}^{-1}$. When $\mathbf{A}$ is not necessarily invertible, the pseudoinverse of $\mathbf{A}$ is denoted by $\mathbf{A}^\dagger$.

$\mathbb{1}_d$ denotes the $d \times d$ identity matrix. If its dimension is implied by the context, then the subscript will be dropped. The vector of all ones is denoted $\mathbf{1}$.

$\mathbf{x} \geq 0$ means that each component of $\mathbf{x}$ is nonnegative.

An $n \times n$ matrix $\mathbf{A}$ is positive semidefinite if $\mathbf{x}^\top \mathbf{A}\mathbf{x} \geq 0$ for all $\mathbf{x} \in \mathbf{R}^n$. Given two positive semidefinite $n \times n$ matrices $\mathbf{A}$ and $\mathbf{B}$, $\mathbf{A} \succeq \mathbf{B}$ means $\mathbf{A} - \mathbf{B}$ is positive semidefinite. In particular, if $\mathbf{A}$ is positive semidefinite then we may write $\mathbf{A} \succeq 0$. The "$\succeq$" relationship is a partial ordering on the semidefinite cone.

If $\mathbf{x}$ is an n-dimensional vector, $\operatorname{diag} \mathbf{x}$ denotes the diagonal matrix with $\mathbf{x}$ on its diagonal. If, on the other hand $\mathbf{A}$ is an $n \times n$ matrix, $\operatorname{diag}(\mathbf{A})$ denotes the vector comprised of the diagonal elements of $\mathbf{A}$. For example

$$\operatorname{diag}\left(\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}\right) = \begin{bmatrix} x_1 & 0 \\ 0 & x_2 \end{bmatrix},$$

$$\operatorname{diag}\left(\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}\right) = \begin{bmatrix} a_{11} \\ a_{22} \end{bmatrix}.$$

Let $\mathbf{M}$ be a matrix partitioned as

$$\mathbf{M} = \begin{bmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{B}^\top & \mathbf{C} \end{bmatrix}$$

The Schur complement of $\mathbf{A}$ in $\mathbf{M}$ is defined to be

$$(\mathbf{M}|\mathbf{A}) = \mathbf{C} - \mathbf{B}^\top \mathbf{A}^\dagger \mathbf{B}$$

and the Schur complement of $\mathbf{C}$ in $\mathbf{M}$ is

$$(\mathbf{M}|\mathbf{C}) = \mathbf{A} - \mathbf{B}\mathbf{C}^\dagger \mathbf{B}^\top$$

The pseudoinverses are replaced by inverses when $\mathbf{A}$ or $\mathbf{C}$ are invertible. Some useful facts on Schur complements are presented in Appendix A.

The expected value of the random variable x will be written as $\mathbb{E}[x]$. If there is confusion about the probability distribution, the expected value with respect to the distribution p will be denoted $\mathbb{E}_p[x]$.

Chapter 2

Mathematical Background

2.1 Basics of Convexity

Beginning with Karmakar's famed interior point algorithm for linear programming [53], rapid advances in algorithms for efficiently operating with convex bodies have rendered convex programs generally no harder to solve than least-squares problems. Once a goal is phrased as a convex set of constraints and costs, it can usually be solved efficiently using a standard set of algorithms. Furthermore, a growing body of work in formulating problems in a convex framework shows a wide applicability across fields. This brief section will provide an overview of the features of convexity that make it such an attractive modeling tool. There are three objects which can be convex: sets, functions, and programs. I will describe what convexity means for each of these in turn.

2.1.1 Convex Sets

A set Ω in Euclidean space is *convex* if it contains all line segments between all points. For every $\mathbf{x}_1$ and $\mathbf{x}_2$ in Ω and every t between 0 and 1, the point $(1-t)\mathbf{x}_1 + t\mathbf{x}_2$ is in Ω. Figure 2-1 shows two convex sets and one nonconvex set. Many familiar spaces are convex. For example, the interior of a square or a disk are both convex subsets of the plane. There are other more abstract convex sets that commonly arise in

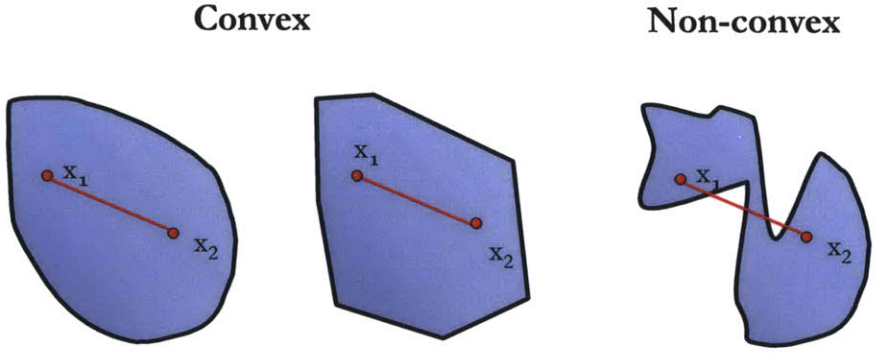

Figure 2-1: **Left and Middle:** Two convex sets. In each set a line segment is drawn between two points in the set and this line never leaves the set. **Right:** A nonconvex set. Two points are shown which cannot be connected by a straight line that doesn't leave the set.

mathematical modeling such as the set of possible covariance matrices of a random process. On the other hand, sets which are not convex abound as well. Neither the set of integers (try, for example, $\mathbf{x}_1 = 1$, $\mathbf{x}_2 = 2$, and $t = 0.5$) nor the set of invertible matrices are convex, and much of the art of convex analysis lies in recognizing when a set is convex.

An important tool for recognizing convex sets is a dictionary of operations that preserve convexity. For example, if $\Omega_1, \ldots, \Omega_m$ are convex, then their intersection $\bigcap_{i=1}^{m} \Omega_i$ is convex. If Ω is convex then any affine transformation of Ω, $\{\mathbf{A}\mathbf{x}+\mathbf{b}|\mathbf{x} \in \Omega\}$ is convex. Furthermore, the premimage of an affine mapping is also convex. That is, if Ω is convex, then so is $\{\mathbf{x}|\mathbf{A}\mathbf{x} + \mathbf{b} \in \Omega\}$.

2.1.2 Convex Functions

The epigraph of a function $f : D \to \mathbb{R}$ is the set

$$\text{epi}(f) = \{(\mathbf{x}, y) \, : \, \mathbf{x} \in D \; y \geq f(\mathbf{x})\} \tag{2.1}$$

A convex function is a real-valued function whose epigraph is convex. That is, if the set of all points lying above the value of the function is convex, then the function is convex (see Figure 2-2). Any linear function is convex as are quadratic forms arising

from matrices with positive eigenvalues. Indeed, a quadratic form $f(\mathbf{x}) = \mathbf{x}^\top \mathbf{Q}\mathbf{x}$ with $\mathbf{Q} = \mathbf{Q}^\top$ is convex if and only if $\mathbf{Q} \succeq 0$. To see this, note that $\mathbf{Q} \succeq 0$ implies $\mathbf{Q} = \mathbf{A}^\top \mathbf{A}$ for some $\mathbf{A}$. Then

$$\mathbf{x}^\top \mathbf{Q}\mathbf{x} = \mathbf{x}^\top \mathbf{A}^\top \mathbf{A}\mathbf{x} = \|\mathbf{A}\mathbf{x}\|^2 \tag{2.2}$$

and thus if $(\mathbf{x}_1, y_1)$ and $(\mathbf{x}_2, y_2)$ are in epi(f) and $t \in [0, 1,]$,

$$\begin{aligned} f(t\mathbf{x}_1 + (1-t)\mathbf{x}_2) &= \|\mathbf{A}(t\mathbf{x}_1 + (1-t)\mathbf{x}_2)\|^2 \\ &\leq t\|\mathbf{A}\mathbf{x}_1\|^2 + (1-t)\|\mathbf{A}\mathbf{x}_2\|^2 \\ &= tf(\mathbf{x}_1) + (1-t)f(\mathbf{x}_2) \\ &\leq ty_1 + (1-t)y_2 \end{aligned} \tag{2.3}$$

proving that $(t\mathbf{x}_1 + (1-t)\mathbf{x}_2, ty_1 + (1-t)y_2) \in \text{epi}_f$. Conversely, if $\mathbf{Q}$ is not positive semidefinite, let $\mathbf{v}$ be a norm 1 eigenvector corresponding to eigenvalue $\lambda < 0$. Then $(-\mathbf{v}, \lambda)$ and $(\mathbf{v}, \lambda)$ are in epi$(\mathbf{Q})$, but $(0, \lambda)$ is not, so f is not convex.

As a consequence of (2.3),when the domain of f is $\mathbb{R}^n$, f is convex if and only if

$$f(t\mathbf{x}_1 + (1-t)\mathbf{x}_2) \leq tf(\mathbf{x}_1) + (1-t)f(\mathbf{x}_2)\,. \tag{2.4}$$

Many other simple functions are not convex including the trigonometric functions sine, cosine, and tangent and most polynomial expressions. An important feature of convex functions is their lack of local minima: if two points minimize a convex function locally, then both points achieve the same function value as do all the points along the line segment connecting them. This is one of the crucial features which makes the minimization of convex functions feasible.

Just as was the case with convex sets, there are a variety of operations that preserve the convexity of functions. For instance, if $f(\mathbf{x})$ is convex then $f(\mathbf{A}\mathbf{x} + \mathbf{b})$ is convex. If $f_1, \ldots, f_n$ are convex, then so is $a_1 f_1 + \cdots + a_n f_n$ for any non-negative scalars a_i.

Less obviously, convex functions are closed under partial maximization. Given a

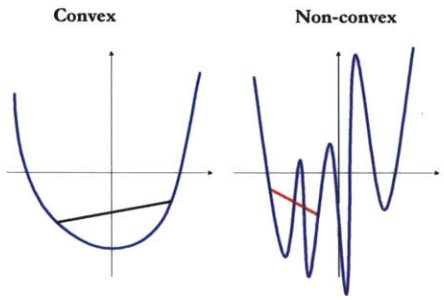

Figure 2-2: **Left:** A convex function. One can readily check that the area above the blue curve contains all line segments between all points. **Right:** The red segment demonstrates that the region above the graph is not convex.

family of convex functions $f_\alpha(\mathbf{x})$ with α in an index set I, $f_m(\mathbf{x}) = \sup_\alpha f_\alpha(\mathbf{x})$ is a convex function. This can be verified by observing

$$\begin{aligned} \text{epi}(f_m) &= \{(\mathbf{x}, y) \,:\, \mathbf{x} \in D \; y \geq f_m(\mathbf{x})\} \\ &= \{(\mathbf{x}, y) \,:\, \mathbf{x} \in D \; y \geq \sup_\alpha f_\alpha(\mathbf{x})\} \\ &= \{(\mathbf{x}, y) \,:\, \mathbf{x} \in D \; y \geq f_\alpha(\mathbf{x}) \, \forall \alpha \in I\} \\ &= \cap_{\alpha \in I} \{(\mathbf{x}, y) \,:\, \mathbf{x} \in D \; y \geq f_\alpha(\mathbf{x})\} \end{aligned} \tag{2.5}$$

which is an intersection of convex sets and must be convex. Two immediate corollaries are that if $f_1, \ldots, f_n$ are convex, then $\max_i f_i(\mathbf{x})$ is convex and, if for all $\mathbf{y}$, $f(\mathbf{x}, \mathbf{y})$ is convex in $\mathbf{x}$, then $\sup_{\mathbf{y}} f(\mathbf{x}, \mathbf{y})$ is convex. It is worth noting that f does not need to be convex in both $\mathbf{x}$ and $\mathbf{y}$ for this to hold.

For differentiable functions that map $\mathbb{R}^n$ to $\mathbb{R}$, convexity may be checked by inspecting derivatives. If f is differentiable, f is convex if and only if $f(\mathbf{y}) \geq f(\mathbf{x}) + \nabla f(\mathbf{x})^\top (\mathbf{y} - \mathbf{x})$ for all $\mathbf{y}$. If f is twice differentiable f is convex if and only if $\nabla^2 f$ is positive semidefinite.

2.1.3 Convex Optimization

We will denote optimization problems

$$\begin{aligned} &\text{minimize} && f(\mathbf{x}) \\ &\text{subject to} && \mathbf{x} \in \Omega \end{aligned} \tag{2.6}$$

by the short hand

$$\begin{aligned} &\min && f(\mathbf{x}) \\ &\text{s.t.} && \mathbf{x} \in \Omega \end{aligned} \tag{2.7}$$

A *convex program* or *convex optimization* seeks to find the minimum of a convex function f on a convex set Ω. As was the case with convex functions, all local minima of convex optimizations are global minima. Remarkably, if testing membership in the Ω and evaluating f can both be performed efficiently, then the minimizer of such a problem can be found efficiently [39]. The most ubiquitous convex program is the least-squares problem which seeks the minimum norm solution to a system of linear equations. In this case f is a convex quadratic function and Ω is Euclidean space.

A fundamental property of convex sets is that they are the intersection of all half-spaces which contain them. That is, if a point $\mathbf{x}$ does not lie in a convex set, then the Euclidean space can be divided into two halves, one half containing $\mathbf{x}$ and the other half containing the convex set. This property suggests that when trying to find an

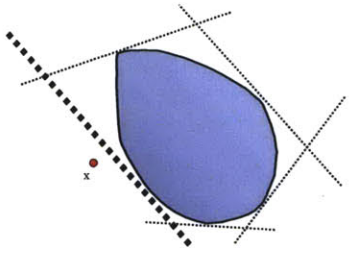

Figure 2-3: The convex set is separated from the points not in the set by half-spaces. The bold dashed line separates the plane into two halves, one containing the point $\mathbf{x}$ and the other containing the convex set.

optimal point in a convex set, one could also search over the set of half-spaces which contain the set (see Figure 2-3). Applying this reasoning to optimization, consider the optimization, called *primal problem*,

$$\begin{aligned} &\text{minimize} \quad f(\mathbf{x}) \\ &\text{subject to} \quad g_j(\mathbf{x}) \leq 0 \quad j = 1, \ldots, J\,. \end{aligned} \tag{2.8}$$

Here f, g_1, $g_2, \ldots, g_j$ are all functions. This is a typical presentation of an optimization problem: the set Ω is the set of all $\mathbf{x}$ for which $g_j(x)$ is nonpositive for all $j = 1, \ldots, J$. In linear programming, both the f and all of the g_j are linear maps.

The *Lagrangian* for this problem is given by

$$\mathcal{L}(\mathbf{x}, \mu) = f(\mathbf{x}) + \sum_{j=1}^{J} \mu_j g_j(\mathbf{x}) \tag{2.9}$$

with $\mu \geq 0$. The μ_j are called *Lagrange multipliers.* In calculus, we searched for values of μ by using $\nabla_{\mathbf{x}} \mathcal{L}(\mathbf{x}, \mu) = 0$. Here, note that solving the optimization is equivalent to solving

$$\min_{\mathbf{x}} \max_{\mu \geq 0} \mathcal{L}(\mathbf{x}, \mu) \tag{2.10}$$

The *Dual Problem* is the resulting problem when the max and min are switched.

$$\max_{\mu \geq 0} \min_{\mathbf{x}} \mathcal{L}(\mathbf{x}, \mu) \tag{2.11}$$

The dual program has many useful properties. First, note that it provides a lower bound of the primal problem. We can show this by appealing to the more general logical tautology

$$\min_{\mathbf{x}} \max_{\mu \geq 0} \mathcal{L}(\mathbf{x}, \mu) \geq \max_{\mu \geq 0} \min_{\mathbf{x}} \mathcal{L}(\mathbf{x}, \mu) \tag{2.12}$$

Indeed, let $f(\mathbf{x}, \mathbf{y})$ be *any* function with two arguments. Then $f(\mathbf{x}, \mathbf{y}) \geq \min_{\mathbf{x}} f(\mathbf{x}, \mathbf{y})$. Taking the max with respect to $\mathbf{y}$ of both sides shows $\max_{\mathbf{y}} f(\mathbf{x}, \mathbf{y}) \geq \max_{\mathbf{y}} \min_{\mathbf{x}} f(\mathbf{x}, \mathbf{y})$.

Now take the min of the right hand side with respect to $\mathbf{x}$ to show that

$$\min_{\mathbf{x}} \max_{\mathbf{y}} f(\mathbf{x}, \mathbf{y}) \geq \max_{\mathbf{y}} \min_{\mathbf{x}} f(\mathbf{x}, \mathbf{y}) \,. \tag{2.13}$$

The dual program is always concave. To see this, consider the *dual function*

$$q(\mu) \equiv \min_{\mathbf{x}} \mathcal{L}(\mathbf{x}, \mu) = \min_{\mathbf{x}} f(\mathbf{x}) + \sum_{j=1}^{J} \mu_j g_j(\mathbf{x}) \tag{2.14}$$

Now, since $\min_{\mathbf{x}}(f(\mathbf{x}) + g(\mathbf{x})) \geq (\min_{\mathbf{x}} f(\mathbf{x})) + (\min_{\mathbf{x}} g(\mathbf{x}))$, we have

$$\begin{aligned} q(t\mu_1 + (1-t)\mu_2) = \min_{\mathbf{x}} & \, t \left(f(\mathbf{x}) + \sum_{j=1}^{J} \mu_{1j} g_j(\mathbf{x}) \right) \\ & + (1-t) \left(f(\mathbf{x}) + \sum_{j=1}^{J} \mu_{2j} g_j(\mathbf{x}) \right) \\ & \geq t q(\mu_1) + (1-t) q(\mu_2) \end{aligned} \tag{2.15}$$

which shows that q is a concave function and hence the dual problem is a convex optimization.

There is a nice graphical interpretation of duality. The *image* is the set of all tuples of numbers $\{(f(\mathbf{x}), g(\mathbf{x}))\}$ for all $\mathbf{x}$. In the image, the optimal value is equal to the minimum crossing point on the y-axis [12]. The dual program seeks to find the half-space which contains the image and which has the greatest intercept with the $f(x)$ axis. As shown in Figure 2-4, the maximum of the dual program is always less than the minimum of the primal program.

Duality is a powerful and widely employed tool in applied mathematics for a number of reasons. First, the dual program is always convex even if the primal is not. Second, the number of variables in the dual is equal to the number of constraints in the primal which is often less than the number of variables in the primal program. Third, the maximum value achieved by the dual problem is often *equal* to the minimum of the primal. One such example when the primal and dual optima are equal is when f and all of the g_j are convex functions and there is a point $\mathbf{x}$ for which $g_j(\mathbf{x})$ is strictly

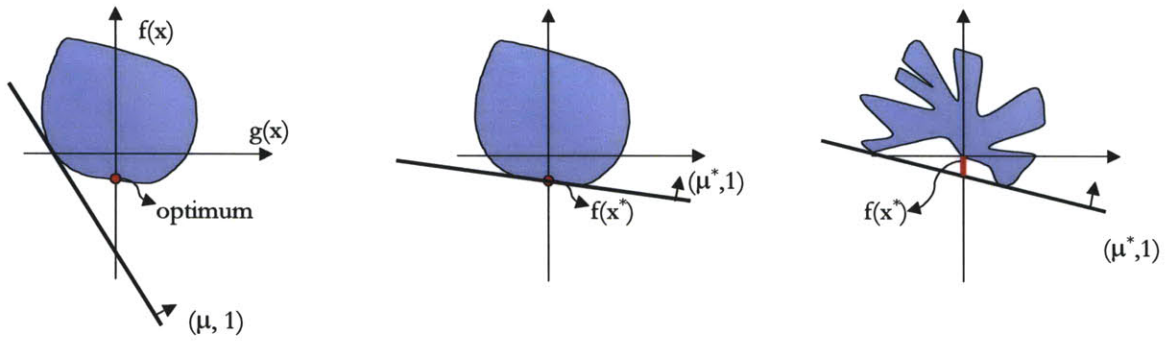

Figure 2-4: The set of possible pairs of $g(\mathbf{x})$ and $f(\mathbf{x})$ are shown as the blue region. **Left:** Any hyperplane which has normal $(\mu, 1)$ intersects the y-axis at the point $f(\mathbf{x}^*) + \mu^\top g(\mathbf{x}^*)$ where $\mathbf{x}^*$ minimizes $\mathcal{L}(\mathbf{x}, \mu)$ with respect to $\mathbf{x}$. **Middle:** A hyperplane whose y intercept is equal to the minimum of $f(x)$ on the feasible set. The dual optimal value is equal to that of the primal **Right:** No hyperplane can achieve the primal optimal value. The discrepancy between the primal and dual optima is called a *duality gap*. The dual optimum value is always a lower bound for the primal.

negative for all j. Finally, if the primal program is not convex or not strictly feasible, it is often possible to bound the *duality gap* between the primal and the dual optimal values. Estimating the duality gap is often difficult and, in many cases, this gap is infinite. However, for many practical problems, several researchers have discovered that one can meticulously bound the duality gap and produce sub-optimal solutions to the primal problems whose cost is only a constant fraction away from optimality. This is the study of *convex relaxations*.

2.2 Convex Relaxations

It is well known that finding the best integer solution to a linear program is NP-HARD. Many of the most successful and popular techniques for dealing with these generally hard problems solve the linear program for the best real valued solution, ignoring the constraint to the set of integers. One gets a lower bound on the optimum, and techniques such as branch and cut or branch and bound can be implemented. This is the most famous example of a convex relaxation. Indeed, this relaxation is well motivated by Lagrangian duality as the program obtained by dropping the integrality

constraint has the same dual program as the primal integer program.

A series of surprising results have been developed over the last ten years using quadratic programming, rather than linear programming, to approach combinatorial problems. The general nonconvex quadratic program is also NP-HARD, and many hard combinatorial problems are naturally expressed as quadratic programs. For example, the requirement that the variable x takes on values 0 or 1 can be expressed by the quadratic constraint $x^2 = x$. The dual program of a general nonconvex quadratic program is a *semidefinite program.* Such optimizations can be solved efficiently using interior point methods [99] among other possible convex optimization techniques. For many structured quadratic constraints, one can actually estimate the worst case duality gap. Moreover, for many problems of interest, there exists a randomized algorithm that produces a vector whose cost is within a constant factor $\gamma < 1$ of the optimal primal value. A randomized algorithm which satisfies such an inequality is called a γ-approximation. There are several examples of hard problems in combinatorial optimization where γ is greater than $1/2$. Indeed, for the famed MAX-CUT relaxation of Goemans and Williamson, $\gamma \leq 0.878$ [37]. This is a great achievement considering that the existence of a polynomial time approximation to MAX-CUT with $\gamma \geq 0.95$ would imply $P = NP$, an equality which the majority of researchers in theoretical computer science think is highly unlikely [43].

In this section I will summarize these techniques providing a unified presentation of the duality structure of nonconvex quadratic programs and how these duals can be used to provide bounds on combinatorial optimization problems. In Section 2.2.1, we will show that the Lagrangian dual of the general nonconvex quadratically constrained quadratic program is a semidefinite program. In Section 2.2.2 we will study how to bound the duality gap and to produce primal feasible points with near optimal cost.

2.2.1 Nonconvex Quadratically Constrained Quadratic Programming

Let us begin with the general nonconvex quadratically constrained quadratic program

$$\begin{array}{ll} \min & \mathbf{x}^\top \mathbf{A}_0 \mathbf{x} + 2\mathbf{b}_0^\top \mathbf{x} + c_0 \\ \text{s.t.} & \mathbf{x}^\top \mathbf{A}_i \mathbf{x} + 2\mathbf{b}_i^\top \mathbf{x} + c_i \leq 0 \quad i = 1, \ldots, K \end{array} \tag{2.16}$$

with $\mathbf{x} \in \mathbb{R}^n$. This problem is again NP-HARD (a recurring theme). It is, of course, well known that this problem is solvable efficiently when the $\mathbf{A}_i$ are positive semidefinite, but in the situation where they are not, we have to rely on more sophisticated techniques for estimating the optimum.

Let's now examine the structure of the Lagrangian dual problem. First, we make a variable substitution to get the equivalent optimization

$$\begin{array}{ll} \min & \mathbf{y}^\top \mathbf{Q}_0 \mathbf{y} \\ \text{s.t.} & \mathbf{y}^\top \mathbf{Q}_i \mathbf{y} \leq 0 \quad i = 1, \ldots, K \\ & y_0^2 = 1 \end{array} \tag{2.17}$$

where y is an $n+1$ dimensional vector and

$$\mathbf{Q}_i = \begin{bmatrix} c_i & \mathbf{b}_i^\top \\ \mathbf{b}_i & \mathbf{A}_i \end{bmatrix}. \tag{2.18}$$

We can think of $\mathbf{y}$ as the original decision variable $\mathbf{x}$ with a 1 stacked on top.

The optimal value of Problem (2.17) is the same as (2.16). Any optimal solution of (2.16) can be turned into a minimizer for (2.17) by setting $\mathbf{x} = [1, \mathbf{x}]$. Since $\mathbf{y}^\top \mathbf{Q} \mathbf{y} = (-\mathbf{y})^\top \mathbf{Q} (-\mathbf{y})$, any optimal solution for (2.17) can be turned into an optimal solution for (2.16) by choosing the solution with $y_0 = 1$.

Problem (2.17) has a particularly elegant dual problem. The Lagrangian for the

reformulated problem is then

$$\mathcal{L}(\mathbf{y}, \mu, t) = \mathbf{y}^\top \mathbf{Q}(\mu, t)\mathbf{y} + t \tag{2.19}$$

where

$$\mathbf{Q}(\mu, t) = \mathbf{Q}_0 + \sum_{i=1}^{K} \mu_i \mathbf{Q}_i - t\delta_{00} \tag{2.20}$$

Minimizing with respect to y, we obtain negative infinity if $\mathbf{Q}(\mu, t)$ has any negative eigenvalues. In turn, we find that the dual function is given by

$$q(\mu, t) = \begin{cases} t & \mathbf{Q}(\mu, t) \succeq 0 \\ -\infty & \text{otherwise} \end{cases} \tag{2.21}$$

and hence the dual problem is

$$\begin{aligned} \max \quad & t \\ \text{s.t.} \quad & \mathbf{Q}_0 + \textstyle\sum_{i=1}^{K} \mu_i \mathbf{Q}_i - t\delta_{00} \succeq 0 \\ & \mu \geq 0 \end{aligned} \tag{2.22}$$

This optimization is called a *semidefinite program* as the search is over the cone of positive semidefinte matrices. The dual can be solved efficiently using interior point methods [99] among other possible convex optimization techniques.

Note that we don't worsen the dual bound by introducing the ancillary variable y_0. To see this, observe that we can break the dual program apart as follows

$$\max_{\mu,t} \min_{\mathbf{y}} \mathcal{L}(\mathbf{y}, \mu, t) = \max_{\mu} \max_{t} \min_{y_0} \min_{y_1,\ldots,y_n} \mathcal{L}(\mathbf{y}, \mu, t) \tag{2.23}$$

For now, ignore the μ maximization and consider the optimization

$$\max_{t} \min_{y_0} \min_{\mathbf{x}} \begin{bmatrix} y_0 \\ \mathbf{x} \end{bmatrix}^\top \begin{bmatrix} c & \mathbf{b}^\top \\ \mathbf{b} & \mathbf{Q} \end{bmatrix} \begin{bmatrix} y_0 \\ \mathbf{x} \end{bmatrix} + t(1 - y_0^2) \tag{2.24}$$

Performing the minimization with respect to $\mathbf{x}$, we either get negative infinity or, if

the matrix is positive semidefinite, we get the Schur complement of the quadratic form

$$\max_t \min_{y_0} y_0^2(-\mathbf{b}^\top \mathbf{Q}^{-1}\mathbf{b} + c - t) + t \tag{2.25}$$

By inspection, the saddle point of this optimization is given when

$$\begin{aligned} t &= -\mathbf{b}^\top \mathbf{Q}^{-1}\mathbf{b} + c \\ y_0^2 &= 1 \end{aligned} \tag{2.26}$$

but that means

$$\begin{aligned} \max_t \min_{y_0} \min_{\mathbf{x}} \begin{bmatrix} y_0 \\ \mathbf{x} \end{bmatrix}^\top \begin{bmatrix} c & \mathbf{b}^\top \\ \mathbf{b} & \mathbf{Q} \end{bmatrix} \begin{bmatrix} y_0 \\ \mathbf{x} \end{bmatrix} + t(1 - y_0^2) = \\ \min_{\mathbf{x}} \begin{bmatrix} 1 \\ \mathbf{x} \end{bmatrix}^\top \begin{bmatrix} c & \mathbf{b}^\top \\ \mathbf{b} & \mathbf{Q} \end{bmatrix} \begin{bmatrix} 1 \\ \mathbf{x} \end{bmatrix} \end{aligned} \tag{2.27}$$

That is, the dual values with or without the additional variable y_0 are the same.

It is instructive to now compute the dual of the dual. A straightforward application of semidefinite programming duality yields the semidefinite program

$$\begin{aligned} \min \quad & \mathrm{Tr}(\mathbf{Q}_0\mathbf{Z}) \\ & \mathrm{Tr}(\mathbf{Q}_i\mathbf{Z}) \leq 0 \quad i = 1, \ldots, K \\ & Z_{00} = 1 \\ & \mathbf{Z} \succeq 0\,. \end{aligned} \tag{2.28}$$

We can show that this relaxation can be derived by dropping refractory nonconvex constraints from the original primal program. This is similar to the relaxations of integer programming that utilize the linear program obtained by relaxing the integrality constraint. In the quadratic case, the constraint that we drop is a constraint on the rank of the matrix $\mathbf{Z}$. To see this, first observe that we have the identity

$$\mathbf{y}^\top \mathbf{Q}\mathbf{y} = \mathrm{Tr}(\mathbf{Q}\mathbf{y}\mathbf{y}^\top) \tag{2.29}$$

and we can prove a simple

Proposition 2.2.1 $\mathbf{Z} = \mathbf{y}\mathbf{y}^\top$ *for some* $\mathbf{y} \in \mathbb{R}^n$ *if and only if* $\mathbf{Z}$ *is positive semidefinite and has rank* 1.

Proof If $\mathbf{Z}$ is positive semidefinite, we can diagonalize $\mathbf{Z} = \mathbf{V}\mathbf{D}\mathbf{V}^\top$ where $\mathbf{V}$ is orthogonal and $\mathbf{D}$ is diagonal. Without loss of generality, $\text{rank}(\mathbf{Z}) = 1$ implies that $\mathbf{D}$ has $d_{11} > 0$ and zeros elsewhere. Then if $\mathbf{v}_1$ is the first column of $\mathbf{V}$,

$$\mathbf{Z} = d_{11}\mathbf{v}_1\mathbf{v}_1^\top = (\sqrt{d_{11}}\mathbf{v}_1)(\sqrt{d_{11}}\mathbf{v}_1)^\top \tag{2.30}$$

Setting $\mathbf{y} = \sqrt{d_{11}}\mathbf{v}$ completes the proof. The converse is immediate. ■

Using this proposition, we can reformulate the original quadtratic program (2.17) as

$$\begin{aligned} \min \quad & \text{Tr}(\mathbf{Q}_0\mathbf{Z}) \\ & \text{Tr}(\mathbf{Q}_i\mathbf{Z}) \leq 0 \quad i = 1, \ldots, K \\ & Z_{00} = 1 \\ & \mathbf{Z} \succeq 0 \\ & \text{rank}(\mathbf{Z}) = 1 \end{aligned} \tag{2.31}$$

The rank constraint is not convex, so a natural convex relaxation would be to drop it. Lo and behold, the resulting optimization is the semidefinite program (2.28).

Unlike the case of integer programming, for structured $\mathbf{Q}_k$ we can actually estimate the worst case duality gap for this relaxation. In special cases, by solving problem (2.28), we can find a real number $\gamma \leq 1$ and use a randomized algorithm to produce a vector $\mathbf{y}$ which is feasible for the optimization (2.17) such that

$$\frac{\mathbb{E}[\mathbf{y}^\top\mathbf{Q}_0\mathbf{y}]}{\mathbf{y}^{\top *}\mathbf{Q}_0\mathbf{y}^*} \geq \gamma\,. \tag{2.32}$$

where $\mathbf{y}^*$ is the optimum solution of the nonconvex problem. A randomized algorithm which satisfies such an inequality is called a γ-approximation. In the next section we

will describe a particular application of this technique to combinatorial optimization.

2.2.2 Applications in Combinatorial Optimization

Consider the special nonconvex quadratic program

$$\min_{\mathbf{x}\in\{-1,1\}^n} \mathbf{x}^\top \mathbf{A}\mathbf{x} \tag{2.33}$$

Where $\mathbf{A}$ is an arbitrary symmetric $n \times n$ matrix. This problem is inherently combinatorial, and not surprisingly, is NP-HARD. We can write this as an nonconvex quadratically constrained quadratic program using the following extended representation

$$\begin{aligned} \min \quad & \mathbf{x}^\top \mathbf{A}\mathbf{x} \\ & x_i^2 = 1 \quad i = 1, \ldots, n \end{aligned} \tag{2.34}$$

The transformation of a set constraint into an algebraic constraint turns out to be the crucial idea. Indeed, there is no apparent duality structure to (2.33) as the only constraint is integrality. Once we have constraints, we can follow our nose and construct the dual program of (2.34)

$$\begin{aligned} \min \quad & \sum_i \lambda_i \\ \text{s.t.} \quad & \mathbf{A} + \operatorname{diag}(\lambda_1, \ldots, \lambda_n) \succeq 0 \end{aligned} \tag{2.35}$$

and we can use semidefinite programming duality again to find a relaxation for (2.33)

$$\begin{aligned} & \min \operatorname{Tr} \mathbf{A}\mathbf{Z} \\ & \text{s.t.} \operatorname{diag}(\mathbf{Z}) = 1 \\ & \mathbf{Z} \succeq 0 \end{aligned} \tag{2.36}$$

This particular relaxation has been studied extensively in the literature, and led to a major breakthrough when Goemans and Williamson showed how to use it for

approximating the maximum cut in a graph. Before we proceed, let us quickly review some terminology from graph theory.

Let $G = (V, E)$ be a graph and let $w : E \to \mathbb{R}$ be an arbitrary function. A *cut* in the graph is a partition of the vertices into two disjoint sets V_1, V_2 such that $V_1 \cup V_2 = V$. Let $F(V_1)$ denote the set of edges which have exactly one node in V_1. By this definition $F(V_1) = F(V_2)$. The weight of the cut is defined the be

$$w(F) = \sum_{f \in F} w(f) \tag{2.37}$$

Consider the optimization

$$\begin{aligned} \mathcal{MC}(G, w) = \quad & \max \quad w(F(U)) \\ & \text{s.t.} \quad U \subset V \end{aligned} \tag{2.38}$$

If the weight function such that $w(e) = 1$ for all $e \in E$, we denote the optimum solution as $\mathcal{MC}(G)$. This optimization is called MAX-CUT and is another of the classic optimization problems which are provably NP-HARD.

We can transform the optimization into an integer quadratic program by deriving the equivalent optimization problem

$$\begin{aligned} \max \quad & \frac{1}{2} \sum_{(u,v) \in E} w_{uv}(1 - x_u x_v) \\ \text{s.t.} \quad & \mathbf{x} \in \{-1, 1\}^{|V|} \end{aligned} . \tag{2.39}$$

The equivalence can be seen as follows: for every set $U \subset V$, let χ^U denote the incidence vector of U in V and set $x(U) = 2\chi^U - 1$. Then if $u \in U$, and $v \in U$, $x_u x_v = 1$ and hence the edge between them is not counted. On the other hand if $u \in U$ and $v \notin U$, $x_u x_v = -1$. It follows that $\frac{1}{2}(1 - x_u x_v) = 1$, and the edge between them is counted with weight w_{uv}.

We can rewrite this optimization in the form of (2.33) by introducing the *Laplacian*

of G. The Laplacian is the $|V| \times |V|$ matrix defined by

$$L_{uv} = \begin{cases} -w_{uv} & (u,v) \in E \\ \sum_{v' \in \mathbf{Adj(v)}} w_{vv'} & u = v \\ 0 & \text{otherwise} \end{cases} \tag{2.40}$$

where $\mathbf{Adj}(v)$ is the set of vertices adjacent to v. It is readily seen that (2.39) is equivalent to

$$\max_{\mathbf{x} \in \{-1,1\}^{|V|}} \frac{1}{4} \mathbf{x}^\top \mathbf{L} \mathbf{x} \tag{2.41}$$

Now we can apply the techniques developed in Section 2.2.1 to the max cut problem to yield the relaxation

$$\begin{aligned} \max \quad & \frac{1}{4} \operatorname{Tr} \mathbf{L}\mathbf{Z} \\ \text{s.t.} \quad & \operatorname{diag}(\mathbf{Z}) = 1 \cdot \\ & \mathbf{Z} \succeq 0 \end{aligned} \tag{2.42}$$

As noted before, we can solve this relaxation using standard algorithms for semidefinite programming.

Thus far we have not addressed the issue of the duality gap at all. We only know that (2.42) is an bound on the maximum cut in the graph. The breakthrough occurs in the algorithm providing a cut, that is, a primal feasible point, from the optimal solution of the relaxation. Consider the following algorithm:

(i) solve (2.42) to yield a matrix $\mathbf{Z}$

(ii) sample a $\mathbf{y}$ from a normal distribution with mean 0 and covariance $\mathbf{Z}$

(iii) return $\mathbf{x} = \operatorname{sign}(\mathbf{y})$

if we define $\operatorname{sign}(0) = 1$, $\mathbf{x}$ will always be a vector with 1's and -1's and hence is primal feasible. As promised, we can characterize the expected quality of it's cut.

Theorem 2.2.2 (Goemans-Williamson) *The algorithm of Goemans and Williamson produces a cut such that*

$$\frac{\mathbb{E}[\mathbf{cut}]}{\mathcal{MC}(G)} \geq \gamma \tag{2.43}$$

with $\gamma \geq 0.87856$.

The proof relies on two lemmas, the first just a bit of calculus

Lemma 2.2.3 *For* $-1 \leq t \leq 1$, $\frac{1}{\pi}\arccos(t) \geq \gamma\frac{1}{2}(1-t)$ *with* $\gamma \geq 0.87856$

The proof of this can be found in [37], or can be immediately observed by plotting arccos.

The second lemma involves the statistics of the random variable $\mathbf{x}$ called a *probit* distribution. Determining the exact probability of drawing a particular $\mathbf{x}$ is practically infeasible to write down in closed form [46], and even approximating the probability would require an intensive Markov Chain Monte Carlo method (see for example [91]). Yet, if we only desire second order information, the situation is considerably better.

Lemma 2.2.4 *If* $\mathbf{y}$ *is drawn randomly from a Gaussian with zero mean and covariance* $\mathbf{Z}$

$$\Pr[\text{sign}(y_i) \neq \text{sign}(y_j)] = \frac{1}{\pi}\arccos(Z_{ij}) \tag{2.44}$$

Proof Let $n = |V|$ and let $\mathbf{e}_i$, $1 \leq i \leq n$ denote the standard basis for $\mathbb{R}^n$. We have

$$\begin{aligned}\Pr[\text{sign}(y_i) \neq \text{sign}(y_j)] &= 2\Pr[y_i > 0,\, y_j < 0] \\ &= 2\Pr[\mathbf{e}_i^\top \mathbf{y} > 0,\, \mathbf{e}_j^\top \mathbf{y} < 0] \\ &= 2\Pr[\mathbf{v}_1^\top \mathbf{w} > 0,\, \mathbf{v}_2^\top \mathbf{w} < 0]\end{aligned} \tag{2.45}$$

where $\mathbf{w}$ is drawn from a Gaussian distribution with zero mean and covariance $\mathbb{1}$ and $\mathbf{v}_1 = \mathbf{Z}^{1/2}\mathbf{e}_i$, $\mathbf{v}_2 = \mathbf{Z}^{1/2}\mathbf{e}_j$. Note that since $\mathbf{Z}$ has ones on the diagonal, the vectors $\mathbf{v}_1$ and $\mathbf{v}_2$ lie on the unit sphere $\mathbb{S}^n \subset \mathbb{R}^n$. Hence, the last probability is the ratio of the volume of the space $\{\mathbf{x} \in \mathbb{S}^n : \mathbf{v}_1^\top \mathbf{x} > 0,\, \mathbf{v}_2^\top \mathbf{x} < 0\}$ to that of $\mathbb{S}^n$. This is the ratio of

the angle between $\mathbf{v}_1$ and $\mathbf{v}_2$ to 2π. Thus we have

$$\begin{aligned} \Pr[\text{sign}(y_i) \neq \text{sign}(y_j)] &= 2\frac{\arccos(\mathbf{v}_1^\top \mathbf{v}_2)}{2\pi} \\ &= \frac{1}{\pi}\arccos(Z_{ij}) \end{aligned} \tag{2.46}$$

which completes the proof. ■

We can now proceed to prove the quality of the Goemans-Williamson relaxation.

Proof [of Theorem 2.2.2] For any edge $e \in E$, let δ_e denote the indicator function for e in the cut. Then the expected value of a cut is

$$\begin{aligned} \mathbb{E}[\mathbf{cut}] = \sum_{e \in E} w_e \mathbb{E}[\delta_e] &= \sum_{i<j} w_{ij} \Pr[\text{sign}(y_i) \neq \text{sign}(y_j)] \\ &= \frac{1}{\pi}\sum_{i<j} w_{ij}\arccos(Z_{ij}) \\ &\geq \gamma\frac{1}{2}\sum_{i<j} w_{ij}(1 - Z_{ij}) \\ &= \frac{1}{4}\gamma \operatorname{Tr}(\mathbf{LZ}) \end{aligned} \tag{2.47}$$

So we have $\mathbb{E}[\mathbf{cut}] \geq \gamma\frac{1}{4}\operatorname{Tr}(\mathbf{LZ}) \geq \gamma\mathcal{MC}(G)$. ■

Remarkably, this technique of sampling from a probit distribution generalizes to a wide class of problems in combinatorial optimization. Notably, Nesterov generalized the results of Goemans and Williamson to a $2/\pi$-approximation for the more general optimization problem [68]

$$\max_{\mathbf{x} \in \{-1,1\}^n} \mathbf{x}^\top \mathbf{A}\mathbf{x} \tag{2.48}$$

with $\mathbf{A} \succeq 0$. His technique rephrases (2.48) as a nonlinear semidefinite program, and then uses the partial order on the semidefinite cone to yield his bound.

Let $\arcsin(\mathbf{M})$ denote the component-wise arcsin of the matrix $\mathbf{M}$.

Theorem 2.2.5 (Nesterov)

$$\max_{\mathbf{x} \in \{-1,1\}^n} \mathbf{x}^\top \mathbf{A}\mathbf{x} = \frac{2}{\pi} \max_{\mathbf{Z} \succeq 0, \operatorname{diag}(\mathbf{Z})=1} \operatorname{Tr}(\mathbf{A}\arcsin(\mathbf{Z})) \tag{2.49}$$

Proof If $\mathbf{X}$ is a matrix of all 1's or -1's then $\frac{2}{\pi}\arcsin(\mathbf{X}) = \mathbf{X}$. Also note that for any $\mathbf{x} \in \{-1, 1\}^n$, $\mathbf{x}\mathbf{x}^\top$ is feasible for the left hand side of the equation. Therefore right hand side is less than or equal to the left hand side. On the other hand, for any positive semidefinite $\mathbf{Z}$ we have seen that for $\mathbf{x}$ drawn from $\mathbf{Z}$ by the random rounding procedure

$$\Pr[x_i \neq x_j] = \frac{1}{\pi}\arccos(Z_{ij}) \tag{2.50}$$

Furthermore, $\mathbb{E}[x_i x_j] = 1 - 2\Pr[x_i \neq x_j]$ and $1 - \frac{2}{\pi}\arccos(t) = \arcsin(t)$ for all $-1 \leq t \leq 1$ Hence for any $\mathbf{Z}$ which is feasible for the left hand side and for the optimal $\mathbf{x}^*$ for the right hand side, we have

$$\mathbf{x}^{*\top}\mathbf{A}\mathbf{x}^* \geq \mathbb{E}_{\mathbf{Z}}[\mathbf{x}^\top \mathbf{A}\mathbf{x}] = \frac{2}{\pi}\operatorname{Tr}(\mathbf{A}\arcsin(\mathbf{Z})) \tag{2.51}$$

showing that right hand side is greater than or equal to the left hand side and completing the proof. ■

To get rid of the arcsin, we use the following property about the partial ordering of positive semidefinite matrices.

Lemma 2.2.6 *If* $\mathbf{Z}$ *is semidefinite and all of its components all between* -1 *and* 1

$$\arcsin(\mathbf{Z}) \succeq \mathbf{Z} \tag{2.52}$$

Proof First note that if $|Z_{ij}| \leq 1$ for all i and j, then the Taylor series for the component-wise arcsin converges. That is,

$$\arcsin(Z_{ij}) = Z_{ij} + \frac{1}{6}Z_{ij}^3 + \frac{3}{40}Z_{ij}^5 + \ldots \tag{2.53}$$

If $\mathbf{A}$ and $\mathbf{B}$ are positive semidefinite then the matrix $\mathbf{C}$ defined as $C_{ij} = A_{ij}B_{ij}$ is also positive semidefinite [45]. Hence we have that $\arcsin(\mathbf{Z}) - \mathbf{Z}$ is a series of positive semidefinite matrices and is hence positive semidefinite. That is $\arcsin(\mathbf{Z}) \succeq \mathbf{Z}$.

■

Theorem 2.2.7 (Nesterov) *Let* $\mathbf{x}^*$ *denote an optimal solution to (2.48). Using the*

random rounding technique produces an approximation to the solution of (2.48) with

$$\frac{\mathbb{E}[\mathbf{x}^\top \mathbf{A}\mathbf{x}]}{\mathbf{x}^{*\top}\mathbf{A}\mathbf{x}^*} \geq \frac{2}{\pi} \tag{2.54}$$

Proof Let $\mathbf{Z}$ be the solution to the relaxed problem. If we draw $\mathbf{x}$ from $\mathbf{Z}$ using the random rounding procedure we get

$$\mathbb{E}[\mathbf{x}^\top \mathbf{A}\mathbf{x}] = \frac{2}{\pi}\operatorname{Tr}(\mathbf{A}\arcsin(\mathbf{Z})) \geq \frac{2}{\pi}\operatorname{Tr}(\mathbf{A}\mathbf{Z}) \geq \frac{2}{\pi}\mathbf{x}^{*\top}\mathbf{A}\mathbf{x}^* \tag{2.55}$$

because $\arcsin(\mathbf{Z}) \succeq \mathbf{Z}$. ■

Much research has been invested into extending these results. For particular classes of positive semidefinite matrix "$\mathbf{A}$," new bounds on other NP-complete problems have been produced. These include a .874-approximation for maximum directed cut, a .941-approximation for maximum 2-satisfiability, a 7/8-approximation for maximum 3-satisfiability, and improved bounds on the number of colors required to color a graph and the number of cuts required to partition a graph [50][52] [31] [38]. All of these algorithms in one way or another use the random rounding technique or a variant thereof.

In some sense, this random rounding is nearly optimal for extracting primal feasible solutions with near optimal cost. Karloff showed that for MAX-CUT, even if one adds an infinite number of valid linear inequalities to the optimization, there exist problems for which the expected value of the random rounding procedure is exactly 0.878 [51]. Feige showed that even if the random rounding process were derandomized to produce an optimal cut from $\mathbf{Z}$, then the 0.878 bound still holds in the worst case [30]. Finding an alternative or generalization of Lagrangian duality for approximating these integer quadratic programs remains an actively pursued area by researchers in combinatorial optimization.

The applications in this thesis are inspired by these original algorithms and extend them to study the operators on high-dimensional spaces. I will apply these tools directly to problems in statistical inference by combining duality tools with Regularization Networks, a powerful representation which makes infinite dimensional

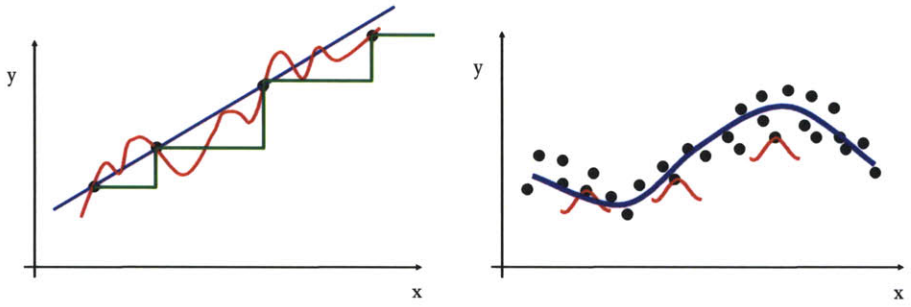

Figure 2-5: **Left:** Given four point, a variety of exact fits are shown. A prior on the function is required to make the problem well-posed. **Right:** Regularization Networks place a "bump" at each observed data point to fit unseen data.

function fitting problems finite.

2.3 Reproducing Kernel Hilbert Spaces and Regularization Networks

One of the most popular and powerful methods for nonlinear function fitting is the optimization based approach with the unfortunately cumbersome name Tikhonov regularization over a Reproducing Kernel Hilbert space. Famous examples of linear least-squares regression, radial basis functions, and support vector machines are all special cases of this framework. Tikhonov regularization both provides a low-dimensional representation of the functions to be fit and, since the resulting optimization is convex, efficient algorithms for determining the function. The resulting function fit can be parameterized with the same number of parameters as training data points, and there are powerful mathematical results showing that, even in the absence of knowledge of the process generating $\mathbf{x}$ and $\mathbf{y}$, the generalization is nearly optimal [19].

In function fitting, one is given pairs of points $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)$ and asked to infer a mapping function that takes as input $\mathbf{x}$ and returns y. What is the best $f : X \rightarrow Y$ that agrees with our data? What is the best f which generalizes to a new data point $\mathbf{x}_{new}$? What are efficient algorithms to approximate this best f with only knowledge of the data?

First we need to define a notion of "best." Suppose that the pointwise cost for making an error is a function is given by a convex cost function $C(f(x), y)$. For example, one might choose the least-squares cost $(f(\mathbf{x})-y)^2$. Support vector machines uses the cost $\max(1-f(\mathbf{x})y, 0)$. This cost assigns no penalty when $|f(\mathbf{x})| > 1$ and has the same sign as y. It assigns a linearly increasing penalty as when these conditions are not satisfied. For a fixed $C(f(x), y)$

$$f^* = \arg\min_f \int_{X,Y} C(f(\mathbf{x}), y) p(\mathbf{x}, y) d\mathbf{x} dy \tag{2.56}$$

would be the ideal function relating $\mathbf{x}$'s and y's. The cost function is called the *risk.* Choosing f according to this rule is called *risk minimization.*

However, if the probability distribution $p(\mathbf{x}, y)$ from which $\mathbf{x}$ and y are drawn is unknown, risk minimization is not possible. Estimating $p(\mathbf{x}, y)$ from sparse data is notoriously difficult and can require an exponential number of samples to get a satisfactory estimate. However, it is often easy to directly estimate

$$f_L^* = \arg\min_f \sum_{i=1}^{L} C(f(\mathbf{x}_i), y_i) \tag{2.57}$$

This cost function is called the *empirical risk.* hoosing f according to this rule is called *empirical risk minimization.*

In the limit of infinite data, the law of large numbers says that the empirical risk converges to the true risk exponentially fast for a fixed function f. However, we are still left with the problem that there is no good way to search over the set of all functions. To fix this, we can restrict f to a specific class of functions $\mathcal{H}$ called the hypothesis space. Then we can try to find

$$f_L^* = \arg\min_{f \in \mathcal{H}} \sum_{i=1}^{L} C(f(\mathbf{x}_i), y_i) \tag{2.58}$$

Even here, there are usually too many available f_L^* which fit the data in the hypothesis space. For example, in the simple case of fitting a linear function to data,

if there are less samples than input dimensions, then there are an infinite set of linear functions that fit the data exactly. We get around this problem by introducing a measure of smoothness $\|f\|$ and try to search for a reasonably smooth function which fits the data

$$f_L^* = \arg\min_{f \in \mathcal{H}} \sum_{i=1}^{L} C(f(\mathbf{x}_i), y_i) + \lambda \|f\|^2 \tag{2.59}$$

The addition of the norm penalty to an optimization is called *Tikhonov Regularization.* By adding the norm penalty, the problem becomes well posed and in many cases has a unique solution. Furthermore, when the norm penalizes complexity, only simple models are optimal. For the remainder of the thesis, Tikhonov Regularization of Empirical Risk Minimization will be referred to simply as Tikhonov Regularization, but the reader should be aware that we are using this term in a rather restricted setting.

When C is a convex function, Tikhonov Regularization is a convex optimization over a (possibly infinite dimensional) space of functions. Our next goal is to establish a class of functions $\mathcal{H}$ and smoothness measures $\|f\|$ for which we can compute f_L^* efficiently, the computation is robust to noise in the data, and the functions f generalize well and are expressive enough to describe real world functional relationships.

The choice of a *Reproducing Kernel Hilbert Space* (RKHS) as a space of candidate functions results in a well-posed problem satisfying all of these requirements. In the next two sections we describe how the Tikhonov Regularization problem is solved for linear regression, and then show how Tikhonov Regularization on an RKHS is solvable using the same techniques as linear regression, allowing for fitting with nonlinear functions that are dense in the continuous functions.

2.3.1 Lessons from Linear Regression

Consider the simple case of fitting the best $f(\mathbf{x}) = \mathbf{w}^\top \mathbf{x}$ for some vector $\mathbf{w}$. Set

$$C(f(\mathbf{x}), y) = (f(\mathbf{x}) - y)^2 \tag{2.60}$$

so that we are solving a least squares fitting problem. Let the smoothness measure on f be the ordinary norm of w

$$\|f\|^2 = \|\mathbf{w}\|^2 \tag{2.61}$$

Then the cost function is

$$\min_w \sum_{i=1}^{L} (\mathbf{w}^\top \mathbf{x}_i - y_i)^2 + \lambda \mathbf{w}^\top \mathbf{w} \tag{2.62}$$

Let $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_L]$ be the matrix with each columns corresponding to each data point in the example set and $\mathbf{y}$ be the vector of labels y_i. $\mathbf{X}$ is called the *data matrix*. Linear regression seeks to find the $\mathbf{w}$ that optimizes

$$\min_w \|\mathbf{X}^\top \mathbf{w} - \mathbf{y}\|^2 + \lambda \mathbf{w}^\top \mathbf{w} \tag{2.63}$$

Taking a derivative with respect to $\mathbf{w}$ and solving for $\mathbf{w}^*$ gives

$$\mathbf{w}^* = (\mathbf{X}\mathbf{X}^\top + \lambda \mathbb{1})^{-1} \mathbf{X}\mathbf{y} \tag{2.64}$$

It is instructive to rewrite this vector as only a function of the Gram matrix of the data. By simple algebra, we can check

$$\mathbf{X}(\lambda \mathbb{1} + \mathbf{X}^\top \mathbf{X}) = (\mathbf{X}\mathbf{X}^\top + \lambda \mathbb{1})\mathbf{X} \tag{2.65}$$

The matrix $\mathbf{G} := \mathbf{X}^\top \mathbf{X}$ is called the *Gram matrix* of the data, and has entries $G_{ij} = \mathbf{x}_i^\top \mathbf{x}_j$. values. $\mathbf{G}$ is $N \times N$ and we only need to know how to compute inner products to compute its entries. Using the identity (2.65), we can rewrite our expression for the optimal $\mathbf{w}$ as

$$\mathbf{w}^* = \mathbf{X}(\lambda \mathbb{1} + \mathbf{G})^{-1} \mathbf{y} \tag{2.66}$$

Furthermore, letting $\mathbf{c} = (\mathbf{G} + \lambda \mathbb{1})^{-1} \mathbf{y}$ we have that the optimal linear function is

given by

$$f^*(\mathbf{x}) = \sum_{i=1}^{L} (c_i \mathbf{x}_i)^\top \mathbf{x} = \sum_{i=1}^{L} c_i (\mathbf{x}_i^\top \mathbf{x}) \tag{2.67}$$

Let us now remark on some of the many useful properties of linear regression. First, all that is required to compute the optimal f is one matrix inversion. This matrix only involved inner products amongst data products. The resulting solution is a linear combination of the data points acting as functions. To compute this function at a test point, we compute a linear combination of inner products between the test point and the data points. Linear functions are somewhat restrictive for general modeling, but fortunately Reproducing Kernel Hilbert Spaces are spaces of nonlinear functions such that Tikhonov Regularization has all of the convenient computational properties of linear regression.

2.3.2 Reproducing Kernel Hilbert Spaces

Let $\Omega \subset \mathbb{R}^d$ be compact. Suppose that $k : \Omega \times \Omega \to \mathbb{R}$ is a symmetric positive definite function in the sense that for all $\mathbf{v}_1, \ldots, \mathbf{v}_J \in \Omega$, $c_1, \ldots c_J \in \mathbb{R}$

$$\sum_{i,j=1}^{J} c_i c_j k(\mathbf{v}_i, \mathbf{v}_j) \geq 0 \,. \tag{2.68}$$

We call such a k a *positive definite kernel.* We will denote the function which maps $\mathbf{u}$ to $k(\mathbf{v}, \mathbf{u})$ by $k(\mathbf{v}, \cdot)$. Some examples of kernels are given in table 2.3.2.

A kernel which satisfies $\mathbf{K}(\mathbf{x}_1, \mathbf{x}_2) = \mathbf{K}(\|\mathbf{x}_1 - \mathbf{x}_2\|)$ is called a radial basis kernel. It turns out that the only RBF Kernels which are positive for all dimensions of the input data are mixtures of Gaussians

$$\mathbf{K}(\mathbf{x}_1, \mathbf{x}_2) = \int \exp(-C\|\mathbf{x}_1 - \mathbf{x}_2\|^2) p(C) dC \tag{2.69}$$

where $p \geq 0$ [86]. The set of such mixtures is equivalent to the set of radial kernels with $\mathbf{K}(\mathbf{x}_1, \mathbf{x}_2) = f(\|\mathbf{x}_1 - \mathbf{x}_2\|^2)$, $(-1)^k \frac{d^k f}{dx^k}(r) \geq 0$ for all $k \geq 0$ and $r \geq 0$. Such an f is called *completely monotonic.*

kernel	functional form
`linear`	$\mathbf{K}(\mathbf{x}_1, \mathbf{x}_2) = \mathbf{x}_1^\top \mathbf{x}_2$
`standard polynomial`	$\mathbf{K}(\mathbf{x}_1, \mathbf{x}_2) = (1 + \mathbf{x}_1^\top \mathbf{x}_2)^d$
`fourier`	$\mathbf{K}(\mathbf{x}_1, \mathbf{x}_2) = \exp(i\mathbf{k}^\top(\mathbf{x}_1 - \mathbf{x}_2))$
radial kernels	functional form
`gaussian`	$\mathbf{K}(\mathbf{x}_1, \mathbf{x}_2) = \exp(-C\|\mathbf{x}_1 - \mathbf{x}_2\|^2)$
`inverse multiquadric`	$\mathbf{K}(\mathbf{x}_1, \mathbf{x}_2) = (\|\mathbf{x}_1 - \mathbf{x}_2\|^2 + C)^{-1/2}$

Table 2.1: Examples of kernel functions

Given J points, $\mathbf{v}_1, \ldots, \mathbf{v}_J \in \Omega$, a consider functions of the form

$$f(\mathbf{u}) = \sum_{j=1}^{J} c_j k(\mathbf{v}_j, \mathbf{u}) \tag{2.70}$$

where k is a positive definite kernel. When k is radial, such an expression is called a *radial basis function.* For a fixed kernel, the set of all such functions over all finite subsets of Ω forms a linear inner product space. First define for all $\mathbf{u}, \mathbf{v} \in \Omega$

$$\langle k(\mathbf{u}, \cdot), k(\mathbf{v}, \cdot) \rangle := k(\mathbf{u}, \mathbf{v}) \,. \tag{2.71}$$

By linearity, this can be immediately extended to inner products of functions of the form (5.9). The completion of this inner product space is called a *Reproducing Kernel Hilbert Space.* This is because the kernel acts as a linear evaluation functional on this Hilbert space. Indeed, defining $L_{\mathbf{u}}$ to be the functional which maps f to $f(\mathbf{u})$ we see that for a function f of the form (5.9)

$$\langle k(\mathbf{u}, \cdot), f \rangle = \left\langle k(\mathbf{u}, \cdot), \sum_{j=1}^{J} c_j k(\mathbf{v}_j, \cdot) \right\rangle = \sum_{j=1}^{J} c_j k(\mathbf{v}_j, \mathbf{u}) = f(\mathbf{u}) = L_{\mathbf{u}}(f) \,. \tag{2.72}$$

Since Ω is compact, each of these functionals $L_{\mathbf{u}}$ must be bounded with a universal constant B

$$|L_u f| \leq B \|f\| \,. \tag{2.73}$$

It is quite easy to show that any Hilbert space in which the evaluation functionals are

bounded, there is a positive definite kernel for which $L_{\mathbf{u}}(f) = \langle k(\mathbf{u}, \cdot), f \rangle$ [105]. This explicit identification of the Hilbert Space structure underlying such kernel models not only helps make proofs trivial, but also results in simple algorithms for solving optimizations with functions in the RKHS as design variables.

Reproducing Kernel Hilbert Spaces have many favorable properties that make their study worthwhile. First, there are a variety of choices of k which make the RKHS dense in $\mathcal{L}^2$ such as the *Gaussian* kernel

$$k(\mathbf{u}, \mathbf{v}) = \exp(-\kappa \|\mathbf{u} - \mathbf{v}\|^2) . \tag{2.74}$$

Several results exist estimating the average distance to functions in $\mathcal{L}^2$ for finite data sets [77, 69, 19]. Second, when the RKHS norm is used as a regularizer or complexity measure for function fitting, the estimated function is insensitive to small changes in the data set such as removing or replacing data points [14]. Finally, it has been shown that kernel functions provide excellent approximations of least-squares regression functions even when the underlying probability distribution that generates the data is unknown [19, 29, 70].

2.3.3 The Kernel Trick and Nonlinear Regression

There is another construction of RKHS directly from the kernel that makes the connection to linear regression explicit. Suppose we lift each data point $\mathbf{x}_i$ with a high (infinite) dimensional vector $\hat{\mathbf{x}}_i := \Phi(\mathbf{x})$ via some prescribed mapping Φ and then solve the linear regression problem with the *lifted data*. We can interpret this as creating a very long list of "features" of each data point and using the features to solve the linear regression problem.

Let $\mathbf{K}$ be a positive definite kernel. By Mercer's theorem

$$K(\mathbf{x}_1, \mathbf{x}_2) = \sum_{i=1}^{\infty} \lambda_i \phi_i(\mathbf{x}_1) \phi_i(\mathbf{x}_2) \tag{2.75}$$

and the sum converges absolutely.

If we lift $\mathbf{x}$ by the rule

$$\hat{\mathbf{x}} = [\sqrt{\lambda_1}\phi_1(\mathbf{x}), \sqrt{\lambda_2}\phi_2(\mathbf{x}), \ldots, \sqrt{\lambda_k}\phi_k(\mathbf{x}), \ldots] \tag{2.76}$$

we find that,

$$\langle \hat{\mathbf{x}}_1, \hat{\mathbf{x}}_2 \rangle = \mathbf{K}(\mathbf{x}_1, \mathbf{x}_2) \tag{2.77}$$

The functions $\phi(\mathbf{x})$ are the promised features. These features can be computed by solving an integral kernel eigenvalue problem, but such computations are not always tractable. Fortunately, we do not need to compute them for most applications. The matrix $\mathbf{K}$ with entries $K_{ij} = \mathbf{K}(\mathbf{x}_i, \mathbf{x}_j)$ is called the kernel matrix. Unless confusion arises, I will abuse notation and use $\mathbf{K}$ for the kernel matrix and $\mathbf{K}(\mathbf{x}_1, \mathbf{x}_2)$ for the kernel function. $\mathbf{K}$ is the lifting of the Gram matrix. By the positivity of the kernel, we know it is a positive semidefinite matrix.

Consider $\alpha = \sum_i c_i \hat{\mathbf{x}}_i$. We can directly compute the norm of $\|\alpha\|_K^2 := \alpha^\top \alpha$ as

$$\|\alpha\|_K^2 = \sum_{i,j} c_i c_j \hat{\mathbf{x}}_i^\top \hat{\mathbf{x}}_j = \mathbf{c}^\top \mathbf{K} \mathbf{c} \geq 0 \tag{2.78}$$

If $\mathbf{x}$ is another point in $\mathbb{R}^D$, then

$$\alpha^\top \hat{\mathbf{x}} = \sum_i c_i \mathbf{K}(\mathbf{x}_i, \mathbf{x}) \tag{2.79}$$

so we can interpret α as a function on the original space X. The kernel trick to tackle nonlinearities is to replace any $\mathbf{x}_1^\top \mathbf{x}_2$ in a linear problem with $\mathbf{K}(\mathbf{x}_1, \mathbf{x}_2)$. This is implicitly lifting the data into an RKHS, and, though it may appear ad hoc, is perfectly rigorous. In particular, replacing the Gram matrix with the kernel matrix, the solution to the linear regression problem with the lifted data will be

$$f(\mathbf{x}) = \sum_{i=1}^{L} c_i \mathbf{K}(\mathbf{x}_i, \mathbf{x}) \tag{2.80}$$

where $\mathbf{c} = (\mathbf{K} + \lambda \mathbb{1})^{-1} \mathbf{y}^\top$. In the next chapter we will give a direct proof that this f

is optimal using a more general theorem called the Representer theorem.

From a practical perspective, any linear function estimation problem which can be phrased in terms of inner products can be "kernelized" into a nonlinear version by replacing all inner products in the problem with kernel evaluations. Thus, the problems of PCA [87], ICA [4] and graph clustering [89] were generalized to their nonlinear counterparts. In the next chapter I will present a straightforward framework for augmenting the power of Tikhonov regularization with constraints on the function outputs that generalizes all of these methods.

Chapter 3

Augmenting Regression with Priors

This chapter lays out the foundation for the remainder of the thesis. At the core is a a powerful cost function (Equation (3.20)) that augments the standard Tikhonov regularization with priors on the labels and the functional form. This cost function can be applied to a vast array of data-driven modeling problems and can be optimized in a principled way via Lagrangian duality.

Our starting point is a new proof of the so-called "Representer Theorem" that shows how almost any norm-regularized cost function we can pose on an RKHS can be transformed into a finite dimensional problem. We will see that this theorem is an immediate consequence of a simple duality argument on the RKHS. By dualizing over the remaining variables in the optimization, we are able to solve joint optimizations over the function that we are fitting, the labels of the data points, and the kernel that defines the Hilbert space.

When learning a mapping, Tikhonov regularization can only take advantage of labelled data, but constraints on the labels of the unlabelled data can also allow us to leverage this unlabelled data. Using Kernel PCA [87] as a motivating example, Section 3.2 shows that augmenting the Tikhonov regularization problem with constraints on the hidden y values can lead to optimization problems where the optimal function includes terms involving the unlabelled data.

Finally, Section 3.3 discusses how to incorporate priors on the functional form of f into learning algorithms. This will result in the problem of selecting the best kernel function for a given problem. The duality argument of Section 3.1 further implies that all of the augmented Tikhonov Regularization problems are convex in the kernel function. When the problem is solvable with a fixed kernel and when a particular optimization problem can be solved over the set of possible kernels that the problem will admit a subgradient algorithm for selecting the optimal function form and the optimal labels for the unlabelled data. This has many implications in the field of "Kernel learning." In particular, whenever the cost function is quadratic or polyhedral and the set of possible kernels defines an affine subset of the positive semidefinite cone then the resulting problem can be phrased as a semidefinite program. The final sections of the chapter discuss several examples of kernel learning. In particular, a new algorithm for computing the best polynomial kernel is presented.

3.1 Duality and the Representer Theorem

In Chapter 2 we discussed a simple optimization over a Reproducing Kernel Hilbert Space and showed that even though the problem was searching over an infinite dimensional space, it could be solved using least-squares. In this section, let us begin with a very general cost Tikhonov regularization problem over an RKHS. Again, we will search for a real valued function f in the RKHS that operates on data $\mathbf{x}$. We will also be interested in optimization over a vector of labels $\mathbf{u}$. The variable $\mathbf{u}$ is a mnemonic for *unlabelled* data to be optimized over, and the variable $\mathbf{y}$ will be reserved for labelled data that are fixed in advance. Let V be *any* cost function on the outputs of f on the data $\{f(\mathbf{x}_j)\}$ and a vector of labels $\mathbf{u}$.

Consider the norm-regularized optimization

$$\min_{f,\mathbf{u}} V(f(\mathbf{x}_1), \ldots, f(\mathbf{x}_N), \mathbf{u}) + \lambda \|f\|_K^2 \tag{3.1}$$

The standard Tikhonov regularized regression problem where all of the labels are

given as $\mathbf{y}$ can be posed in this form by letting V be given as

$$V(f(\mathbf{x}_1),\ldots,f(\mathbf{x}_N),\mathbf{y}) = \sum_{i=1}^{N}(f(\mathbf{x}_i)-u_i)^2 + \delta(\mathbf{u}-\mathbf{y}) \tag{3.2}$$

the delta function removes the variable $\mathbf{u}$ from the optimization and replaces it with the constant vector $\mathbf{y}$. This is equivalent to adjoining the equality constraint $\mathbf{u} = \mathbf{y}$ to the optimization (3.1).

The following theorem has many proofs and is at the heart of why the RKHS framework is useful. Chapter 2 presented a circuitous proof by appealing to linear regression. Other more direct proofs can be found in a variety of sources including, for example, [35, 88].

Theorem 3.1.1 *[Representer Theorem] If f is an optimal solution to (3.1), then f can be expressed in the representation*

$$f(\mathbf{x}) = \sum_{i=1}^{N} c_i k(\mathbf{x}_i,\mathbf{x}) \tag{3.3}$$

This theorem transforms the search over a possibly infinite dimensional space into the search for a set of N real numbers. The proof I present here will serve as a prequel to the use of duality in learning in the remainder of the chapter. The representer form will be an immediate consequence of the solution method. By adding redundant constraints to the cost function, I will employ Lagrangian duality to eliminate f in the RKHS. The resulting Lagrange multipliers will precisely be the coefficients of the expansion (3.3). The proof rests on the following theorem that is proven in Appendix B.

Theorem 3.1.2 *Let $\mathcal{V}$ be a real inner product space and let $\mathbf{w}_1,\ldots,\mathbf{w}_N \in \mathcal{V}$, $\mathbf{a} \in \mathbb{R}^N$. Let $\mathbf{W}$ be the Gram matrix of the $\mathbf{w}_j$.*

(i) The equality constrained norm minimization problem

$$\begin{aligned} \min_{\mathbf{v}\in\mathcal{V}} \quad & \langle \mathbf{v}, \mathbf{v} \rangle \\ s.t. \quad & \langle \mathbf{v}, \mathbf{w}_i \rangle = a_i \quad \textit{for } i = 1, \ldots, N \end{aligned} \tag{3.4}$$

has an associated dual program

$$\max_{\alpha} -\alpha^\top \mathbf{W}\alpha + 2\alpha^\top \mathbf{a} \tag{3.5}$$

which is an unconstrained convex quadratic program. The primal optimal value is equal to the dual optimal value.

(ii) Suppose the optimal value of the primal-dual pair is finite. Then the set of dual optimal solutions is given by

$$\mathcal{D} := \{\alpha \in \mathbb{R}^N \,:\, \mathbf{W}\alpha = \mathbf{a}\} \tag{3.6}$$

and the set of primal optimal solutions is given by

$$\mathcal{P} := \{\sum_{i=1}^{N} \alpha_i \mathbf{w}_i \,:\, \alpha \in \mathcal{D}\} \tag{3.7}$$

With this theorem in hand, let us proceed to a

Proof [of the Representer Theorem] First, introduce a new variable $\mathbf{z} \in \mathbb{R}^N$ and add the constraints

$$z_i = f(\mathbf{x}_i) = \langle f, k(\mathbf{x}_i, \cdot) \rangle_K \tag{3.8}$$

to the primal problem for $i = 1, \ldots N$. This results in the optimization

$$\begin{aligned} \min_{f,\mathbf{z},\mathbf{u}} \quad & V(\mathbf{z}, \mathbf{u}) + \lambda \langle f, f \rangle_K \\ \text{s.t.} \quad & z_i = \langle f, k(\mathbf{x}_i, \cdot) \rangle_K \quad \text{for } i = 1, \ldots, N \end{aligned} \tag{3.9}$$

Now we can dualize over the f variable alone. Since the first summation does not

depend of f, we can ignore it when optimizing over f and consider the inner-most minimization

$$\begin{aligned} \min_{f} \quad & \langle f, f \rangle_K \\ \text{s.t.} \quad & z_i = \langle f, k(\mathbf{x}_i, \cdot) \rangle_K \quad \text{for } i = 1, \ldots, N \end{aligned} \tag{3.10}$$

This is precisely an equality constrained norm minimization on the RKHS of the form (3.4). By Theorem 3.1.2, the dual optimization is given by the unconstrained quadratic program

$$\max_{\mathbf{c}} 2\mathbf{c}^\top \mathbf{z} - \mathbf{c}^\top \mathbf{K} \mathbf{c} \tag{3.11}$$

This is an unconstrained quadratic program. The set of dual optimal solutions is given by the set of solutions to $\mathbf{K}\mathbf{c} = \mathbf{z}$. When $\mathbf{z}$ is in the range of $\mathbf{K}$, the optimal cost is given by $\mathbf{z}^\top \mathbf{K}^\dagger \mathbf{z}$. Plugging this into the original problem gives an optimization free of both f and $\mathbf{c}$.

$$\begin{aligned} \min_{\mathbf{z},\mathbf{u}} \quad & V(\mathbf{z}, \mathbf{u}) + \lambda \mathbf{z}^\top \mathbf{K}^\dagger \mathbf{z} \\ \text{s.t.} \quad & \mathbf{z} \in \text{Ran}(\mathbf{K}) \end{aligned} \tag{3.12}$$

Given the optimal $\mathbf{z}^*$, we can compute

$$\begin{aligned} \mathbf{c}^* &= \mathbf{K}^\dagger \mathbf{z}^* \\ f^* &= \sum_{i=1}^{N} c_i^* k(\mathbf{x}_i, \cdot) \end{aligned} \tag{3.13}$$

■

There are two useful alternative formulations of the Tikhonov regularization problem. First we can substitute the value $\mathbf{K}\mathbf{c}$ in for $\mathbf{z}$ and eliminate the $\mathbf{z}$ variable. This results in the standard representer form that appears in the literature

$$\min_{\mathbf{c},\mathbf{u}} V(\mathbf{K}\mathbf{c}, \mathbf{u}) + \lambda \mathbf{c}^\top \mathbf{K} \mathbf{c} \tag{3.14}$$

Secondly, to avoid the numerical difficulties involved in the computation of pseu-

doinverses, we can employ the Schur complement lemma to yield an equivalent optimization

$$\begin{aligned} \min_{\mathbf{z},\mathbf{u},t} \quad & V(\mathbf{z},\mathbf{u}) + \lambda t \\ \text{s.t.} \quad & \begin{bmatrix} t & \mathbf{z}^\top \\ \mathbf{z} & \mathbf{K} \end{bmatrix} \succeq 0 \end{aligned} \tag{3.15}$$

The equivalence follows immediately from the following simple whose proof can be found in Appendix A

Lemma 3.1.3 *Let* $\mathbf{A} \succeq 0$ *be* $n \times n$, $\mathbf{x} \in \mathbb{R}^n$, *and* $t \in \mathbf{R}$. *Then*

$$\begin{bmatrix} t & \mathbf{x}^\top \\ \mathbf{x} & \mathbf{A} \end{bmatrix} \succeq 0 \tag{3.16}$$

if and only if $\mathbf{x}$ *is in the range of* $\mathbf{A}$ *and* $\mathbf{x}^\top \mathbf{A}^\dagger \mathbf{x} \leq t$.

This reformulation shows that the arbitrary learning problem (3.1) can be formulated such that the kernel matrix appears only in a matrix inequality of the form of (3.15). This inequality is convex in the matrix $\mathbf{K}$, and since each entry of $\mathbf{K}$ is simply an evaluation of a kernel function, the optimization is convex in the kernel function. In particular, if the optimization is jointly convex in $\mathbf{z}$ and $\mathbf{y}$, then it it is convex in both the outputs and the kernel function, and can be solved efficiently. In the Section 3.3 we present a general algorithm for solving this problem. We show that in the case that the cost and $\mathcal{K}$ are both semidefinite representable, then the kernel learning problem can be solved with semidefinite programming.

As an interesting application of the reasoning used in the proof of Theorem 3.1.1, we can show that unseen or unconstrained data has no effect on the optimal f. Indeed, consider the norm minimization problem with $N + M$ data points, the first N fixed, and the last M free:

$$\begin{aligned} \min_{f, z_{N+1}, \ldots, z_{N+M}} \quad & \langle f, f \rangle_K \\ \text{s.t.} \quad & z_i = \langle f, k(\mathbf{x}_i, \cdot) \rangle_K \quad \text{for } i = 1, \ldots, N+M \end{aligned} \tag{3.17}$$

Once again, consider the dual problem with respect to the variables $f, z_{N+1}, \ldots, z_{N+M}$ by forming the Lagrangian with z_i fixed for $i = 1, \ldots, N$.

$$\mathcal{L}(f, z_{N+1}, \ldots, z_{N+M}, \mathbf{c}) = \langle f, f \rangle - 2\langle f, \sum_{i=1}^{N+M} c_i k(\mathbf{x}_i, \cdot) \rangle + \sum_{i=1}^{N+M} c_i z_i \tag{3.18}$$

Let us first minimize jointly over z_i for $i = N+1, \ldots, N+M$. In this case, when any of the corresponding $c_i \neq 0$ for $i = N+1, \ldots, N+M$, the Lagrangian is unbounded below. Therefore, c_i must equal zero in the kernel expansion and the unconstrained data points have no influence on the optimal f. There is again no duality gap, because at the dual optimal $\mathbf{c}$, the primal optimal f has norm

$$\| \sum_{i=1}^{N+M} c_i^* k(\mathbf{x}_i, \cdot) \|^2 = \mathbf{c}^{*\top} \mathbf{K} \mathbf{c}^* \tag{3.19}$$

which is the dual optimal value. In the next section, we will demonstrate how to leverage unlabelled data by constraining the labels of u_i at the unlabelled points.

3.2 Augmenting Regression with Priors on the Output

Rather than using a very general cost V, it will be useful to restrict attention to a more narrow generalization of the least-squares cost of Chapter 2. By a *cost function*, I mean a real valued function $C : \mathbb{R} \times \mathbb{R} \mapsto \mathbb{R}$ such that $C \geq 0$ and for every a, there exists a $b(a)$ such that $C(a, b(a)) = 0$ and for every b there exists an $a(b)$ such that $C(a(b), b) = 0$. Certainly, the least-squares cost satisfies these properties. In Chapter 4, we will encounter the *hinge loss* of the support vector machine [101] which is a also cost function of this form.

Consider the following special case of (3.1)

$$\min_{f, \mathbf{u}} \frac{1}{N} \sum_{i=1}^{N} C(f(\mathbf{x}_i), u_i) + \lambda \|f\|_K^2 + \mathcal{S}(\mathbf{u}) \tag{3.20}$$

where C is any cost function and $\mathcal{S}$ is any extended-real valued function on $\mathbb{R}^n$. Following the same reasoning as in the proof of the Representer Theorem, we must have that the optimal f has the form of Equation (3.3). This is true for *any* function $\mathcal{S}$.

A "prior on output" is just a choice of the function $\mathcal{S}$. The most trivial example is in the case of the standard regression where for each $\mathbf{x}_i$ we are given labels $u_i = y_i$. Here

$$\mathcal{S}(\mathbf{y}) := \begin{cases} 0 & u_i = y_i \, \forall i \\ \infty & \text{otherwise} \end{cases} \tag{3.21}$$

In this case, we can directly minimize over $\mathbf{y}$ by plugging these values into the cost function. Then we can in turn solve for f.

Moving beyond this trivial cost, $\mathcal{S}$, we shall see that a surprising number of well known algorithms can be cast in the form of Equation (3.20). Furthermore, Chapters 4– 6 will present new novel applications of this seemingly simple framework.

3.2.1 Least-Squares Cost

When the cost is the least-squares cost, we can directly minimize it respect to f and are then left with an optimization only over the labels $\mathbf{u}$ and the kernel matrix $\mathbf{K}$. Beginning with the optimization

$$\begin{aligned} \min_{f,\mathbf{z}} \quad & \sum_{i=1}^{L} (z_i - u_i)^2 + \lambda \|f\|_K^2 \\ \text{s.t.} \quad & \langle f, k(\mathbf{x}_i, \cdot) \rangle_K = z_i \end{aligned} \tag{3.22}$$

we can construct the joint dual problem over f and $\mathbf{z}$. Again constructing a Lagrangian

$$\mathcal{L}(f, \mathbf{z}, \mathbf{c}) = \sum_{i=1}^{N} (z_i - u_i)^2 + \lambda \langle f, f \rangle_K - 2\lambda \langle f, \sum_{i=1}^{N} c_i k(\mathbf{x}_i, \cdot) \rangle + \lambda \sum_{i=1}^{N} c_i z_i \tag{3.23}$$

Minimizing over f yields the representer form (3.11). To minimize over $\mathbf{z}$, note that we can minimize over each z_i individually. By differentiating, we find

$$\min_z (z-u)^2 + 2\lambda cz = -\lambda^2 c^2 + 2\lambda cu \tag{3.24}$$

Plugging this into the Lagrangian gives the dual problem

$$\lambda \max_{\mathbf{c}} \quad -\lambda \mathbf{c}^\top \mathbf{c} + 2\mathbf{u}^\top \mathbf{c} - \mathbf{c}^\top \mathbf{K}\mathbf{c} \tag{3.25}$$

This can be readily solved to give the optimal solution

$$\mathbf{c} = (\mathbf{K} + \lambda \mathbb{1})^{-1}\mathbf{u} \tag{3.26}$$

Plugging this back into the dual problem yields the following expression for the cost of the optimal function f in the regularized least-squares problem

$$\mathbf{C}_T(\mathbf{u}, \mathbf{K}) := \lambda \mathbf{u}^\top (\mathbf{K} + \lambda \mathbb{1})^{-1}\mathbf{u} \tag{3.27}$$

called the *Tikhonov cost.* Equation (3.27) will be a focus throughout. It is jointly convex in $\mathbf{u}$ and $\mathbf{K}$ and no matter what cost $S(\mathbf{u})$ we add to the Tikhonov loss, we can always solve for f to produce a term of this form for $\mathbf{y}$. That is, when we choose C to be the least-squares cost, Equation 3.20 can be written as an optimization only over $\mathbf{u}$

$$\min_{\mathbf{u}} \lambda \mathbf{u}^\top (\mathbf{K} + \lambda \mathbb{1})^{-1}\mathbf{u} + \mathcal{S}(\mathbf{u}) \tag{3.28}$$

If we want to determine f, we may take the optimal $\mathbf{y}$ and let $\mathbf{c} = (\mathbf{K} + \lambda \mathbb{1})^{-1}\mathbf{y}$. Then the optimal f is given by the kernel expansion

$$f(\mathbf{x}) = \sum_{i=1}^{N} c_i K(\mathbf{x}_i, \mathbf{x}) \tag{3.29}$$

3.2.2 The Need for Constraints

In the case that some of the y_i labels are withheld, the standard Tikhonov regularization problem cannot make use of any of the unlabelled data points in the kernel expansion. Indeed, if a label y_k is withheld for a data point $\mathbf{x}_k$, we can still include it in the loss function and solve for the optimal label u_k. Without loss of generality, we may assume $k = 1$.

$$\begin{aligned} \min_{f,u_1} \quad & \sum_{i=1}^{N} C(f(\mathbf{x}_i), u_i) + \lambda \|f\|_K^2 \\ \text{s.t.} \quad & u_i = y_i \qquad i = 2, \ldots, N \end{aligned} \tag{3.30}$$

Since u_1 can always be set to force $C(f(\mathbf{x}_1), u_1) = 0$, the outputs of f are only constrained at $\mathbf{x}_i$ for $i = 2, \ldots, N$. As reasoned in Section 3.1, this means that $c_1 = 0$ in the representer form.

For the least-squares cost, we can derive this result directly by plugging in the representer form for $f(\mathbf{x})$ and showing that $c_k = 0$. First we solve the optimization

$$\lambda \min_{u_1} \mathbf{u}^\top (\mathbf{K} + \lambda \mathbb{1})^{-1} \mathbf{u} \tag{3.31}$$

for u_1. To do so, let $\mathbf{y}_2$ denote the vector of labels 2 to N. Partition the kernel matrix around the first entry as

$$\mathbf{K} = \begin{bmatrix} K_{11} & \mathbf{K}_{12} \\ \mathbf{K}_{21} & \mathbf{K}_{22} \end{bmatrix} \tag{3.32}$$

where K_{11} is 1×1, $\mathbf{K}_{12} = \mathbf{K}_{21}^\top$ is $1 \times (N-1)$ and K_{22} is $(N-1) \times (N-1)$. Let $\mathbf{Q} = (\mathbf{K} + \lambda \mathbb{1})^{-1}$ be partitioned in the same way as $\mathbf{K}$

$$\mathbf{Q} = \begin{bmatrix} Q_{11} & \mathbf{Q}_{12} \\ \mathbf{Q}_{21} & \mathbf{Q}_{22} \end{bmatrix} \tag{3.33}$$

Then, ignoring the constant multiple λ, we would like to solve

$$\min_{u_1} u_1^2 Q_{11} + 2u_1 \mathbf{Q}_{12}\mathbf{y}_2 + \mathbf{y}_2^\top \mathbf{Q}_{22}\mathbf{y}_2 \tag{3.34}$$

since the last term is not a function of u_1, we can ignore it. Using the least squares formula and the form of the inverse of a partitioned matrix found in Appendix A, we find

$$\begin{aligned} u_1 &= -\frac{1}{Q_{11}}\mathbf{Q}_{12}\mathbf{y}_2 \\ &= -(\mathbf{K}|\mathbf{K}_{22})(-(\mathbf{K}|\mathbf{K}_{22})^{-1}\mathbf{K}_{12}\mathbf{K}_{22}^{-1})\mathbf{y}_2 \\ &= \mathbf{K}_{12}\mathbf{K}_{22}^{-1}\mathbf{y}_2 \end{aligned} \tag{3.35}$$

Note that if f_2^* is the minimizer of the optimization trained on $\mathbf{x}_2, \ldots, \mathbf{x}_L$, then $u_1 = f_2^*(\mathbf{x}_1)$. Now we can plug this form into the function learned from $\mathbf{x}_1, \ldots, \mathbf{x}_L$

$$c_1 = Q_{11}u_1 + \mathbf{Q}_{12}\mathbf{y}_2 = 0 \tag{3.36}$$

as claimed.

The preceding reasoning shows that one may consider Tikhonov regularization as an optimization over labelled and unlabelled examples. The functional representation does not make use of the unlabelled data, but the hidden labels and the representer form can be optimized simultaneously. On the other hand, if we can yield functional representations that use the unlabelled data if we constrain the outputs at those data points away from their automatic solution in Tikhonov regularization. This will be illustrated in the sequel.

3.2.3 Priors on the Output

Let us now present our first example of how constraints allow for representer forms that exploit the unlabelled data. Consider the cost function

$$\mathcal{S}(\mathbf{u}) = \begin{cases} 0 & \frac{1}{N}\sum_{i=1}^{N} u_i^2 = 1 \\ \infty & \text{otherwise} \end{cases} \tag{3.37}$$

This cost enforces the constraint that the set of labels must have empirical variance 1. Consider the situation when there are *no labels given at all.* If we adjoin the preceding cost to the Tikhonov cost, we can form the optimization problem

$$\begin{aligned} \min_{f,\mathbf{u}} \quad & \sum_{i=1}^{N} (f(\mathbf{x}_i) - u_i)^2 + \lambda \|f\|_k^2 \\ \text{s.t.} \quad & \frac{1}{N} u_i^2 = 1 \end{aligned} \tag{3.38}$$

Using the form for the least squares cost, this reduces to

$$\begin{aligned} \min_{\mathbf{u}} \quad & \lambda \mathbf{u}^\top (\mathbf{K} + \lambda \mathbb{1})^{-1} \mathbf{u} \\ \text{s.t.} \quad & \mathbf{u}^\top \mathbf{u} = N, \end{aligned} \tag{3.39}$$

This is an eigenvalue problem, the optimal solution of which is the greatest eigenvector of $\mathbf{K}$ scaled so that it has norm $\sqrt{N}$. In particular, the optimal $\mathbf{u}$ is not 0. This is because we only allow non-zero solutions by enforcing the variance constraint. Having computed the optimal $\mathbf{u}$, we can, again solve for the optimal function f. Since $\mathbf{u}$ is nonzero, the kernel expansion will include terms from all of the data even though no labels were given.

Let us now consider the situation where we are looking for d functions $f^{(i)}$, $i = 1, \ldots, d$, at once. Again, we will provide no labels for the data but will supply a variance constraint. Let $\mathbf{f}(\mathbf{x})$ denote the vector valued function from $\mathbb{R}^D$ to $\mathbb{R}^d$ with ith component $f^{(i)}$. We will search for a matrix of labels $\mathbf{U}$ with U_{ij} denoting the hidden label for data point $\mathbf{x}_j$ and component i. $\mathbf{U}_j$ will denote the vector of labels

for point $\mathbf{x}_j$. We can form the cost function

$$\begin{aligned} &\min_{\mathbf{f},\mathbf{U}} \quad \sum_{i=1}^{N} \|\mathbf{U}_i - \mathbf{f}(\mathbf{x}_i)\|^2 + \lambda \sum_{i=1}^{d} \|f^{(i)}\|_K^2 \\ &\text{s.t. } \frac{1}{N}\mathbf{U}\mathbf{U}^\top = \mathbb{1}, \end{aligned} \tag{3.40}$$

where the norm of each dimension of $\mathbf{f}$ is penalized individually and the labels are required to have the identity matrix as their empirical covariance. Substituting in the representer form for each component of $\mathbf{f}$, we can define a matrix of weights $\mathbf{C}$ such that

$$f^{(i)}(\mathbf{x}) = \sum_{i=1}^{N} C_{ij} k(\mathbf{x}_i, \mathbf{x}) \tag{3.41}$$

The optimization (3.40) can then be rewritten as:

$$\min_{\mathbf{C},\mathbf{U}} \|\mathbf{U} - \mathbf{C}\mathbf{K}\|_F^2 + \lambda \operatorname{Tr} \mathbf{C}\mathbf{K}\mathbf{C}^\top \tag{3.42}$$

$$\text{s.t. } \frac{1}{N}\mathbf{U}\mathbf{U}^\top = \mathbb{1}, \tag{3.43}$$

where $\|\cdot\|_F$ is the Frobenius norm, $\mathbf{K}$ is the kernel matrix of the $\mathbf{x}_j$.

Rewriting (3.42) as

$$\min_{\mathbf{C},\mathbf{U}} \sum_{i=1}^{d} \begin{bmatrix} \mathbf{U}_i \\ \mathbf{C}_i \end{bmatrix}^\top \begin{bmatrix} \mathbb{1} & -\mathbf{K} \\ -\mathbf{K} & \mathbf{K}^2 + \lambda \mathbf{K} \end{bmatrix} \begin{bmatrix} \mathbf{U}_i \\ \mathbf{C}_i \end{bmatrix} \tag{3.44}$$

$$\text{s.t. } \frac{1}{N}\mathbf{U}\mathbf{U}^\top = \mathbb{1}, \tag{3.45}$$

where $\mathbf{C}_i$ is the transpose of the ith row of $\mathbf{C}$ and $\mathbf{U}_i$ is the transpose of the ith row of $\mathbf{X}$, we may minimize over $\mathbf{C}$ to find

$$\mathbf{C}_i^* = (\mathbf{K} + \lambda \mathbb{1})^{-1} \mathbf{U}_i \tag{3.46}$$

plugging this optimal value back into the cost function yields a minimization only

over $\mathbf{U}$

$$\min_{\mathbf{U}} \sum_{i=1}^{d} \mathbf{U}_i^\top (\mathbf{K} + \lambda \mathbb{1})^{-1} \mathbf{U}_i \tag{3.47}$$

$$\text{s.t. } \frac{1}{N} \mathbf{U}\mathbf{U}^\top = \mathbb{1}, \tag{3.48}$$

Now the optimal $\mathbf{U}_i^*$ are the d largest eigenvalues of the kernel $\mathbf{K}$. Let $\gamma_1, \ldots, \gamma_d$ be the largest eigenvalues of $\mathbf{K}$ and $\mathbf{v}_1, \ldots, \mathbf{v}_d$ be the corresponding orthonormal set eigenvectors. It follows that the optimal $\mathbf{C}_i^*$ are given by

$$\mathbf{C}_i^* = \frac{\sqrt{N}}{\gamma_i + \lambda} v_i \tag{3.49}$$

Substituting this $\mathbf{C}_i^*$ into the representer form gives us a solution for $\mathbf{f}^*$

$$f^{(i)}(\mathbf{x}) = \frac{\sqrt{N}}{\gamma_i + \lambda} \sum_{j=1}^{N} v_{ij} K(\mathbf{x}_j, \mathbf{x}) \tag{3.50}$$

To summarize, finding the optimal $\mathbf{U}^*$ reduces to extracting the d largest eigenvectors of the kernel matrix $\mathbf{K}$. The rows of the optimal $\mathbf{C}^*$ are scaled versions of the rows of $\mathbf{U}^*$.

This first example of regression with an output prior provides a function learning interpretation of the kernel principal components algorithm (KPCA) of Schölkopf et al.[87]. KPCA uses the kernel trick of Chapter 2 to "kernelize" the standard principal component analysis problem from pattern recognition. Instead of operating on the data points $\mathbf{x}_j$, they perform KPCA of the covariance matrix of the lifted data $\hat{\mathbf{x}}_j$ in a Reproducing Kernel Hilbert Space. KPCA returns the functions $h_1, \ldots, h_d$ given by

$$h_i(\mathbf{x}) = \frac{1}{\sqrt{\gamma_i}} \sum_{j=1}^{N} v_{ij} k(\mathbf{x}_j, \mathbf{x}) \tag{3.51}$$

from which we see that the f_i^* are multiples of the solution to the kpca problem

$$f_i^* = \sqrt{\frac{\gamma_i}{(\gamma_i + \lambda)^2}}\ h_i\,. \tag{3.52}$$

This interpretation reveals that KPCA looks for a smooth function f that projects the sequence of observations $\mathbf{X}$ to a low-dimensional sequence $\mathbf{U}$ so that the dimensions of $\mathbf{U}$ are orthogonal to each other and have unit sample variance. By placing an additional prior on the hidden sequence $\mathbf{U}$, we can refine this algorithm to take in to account a wide range of priors. In particular, in Chapter 5 we can constrain the $\mathbf{U}$ to have linear Gaussian dynamics and in Chapter 6 we will search for unit norm functions that are zero on all of the given examples.

3.2.4 Semi-supervised and Unsupervised Learning

A semi-supervised, or *transductive*, learning problem is one where we are given N data points $\mathbf{x}_1, \ldots, \mathbf{x}_N$ with labels for those i that are in a subset $S \subset \{1, \ldots, N\}$. For our purposes, all semi-supervised algorithms are of the form

$$\begin{aligned} \min_{f,\mathbf{u}} \quad & \frac{1}{N}\sum_{i=1}^{N} C(f(\mathbf{x}_i), u_i) + \lambda \|f\|_K^2 + \mathcal{S}(\mathbf{u}) \\ \text{s.t.} \quad & u_i = y_i \text{ for } i \in S \end{aligned} \tag{3.53}$$

with $\mathcal{S}$ a given extended real-valued function.

Unsupervised learning problems are those where we are given N data points $\mathbf{x}_1, \ldots, \mathbf{x}_N$ and no labels at all. For example, the optimization that re-derives kernel PCA is an unsupervised learning problem. In this case, we must guarantee that $\min_u \mathcal{S}(\mathbf{u}) \leq \mathcal{S}(0)$ or else the optimal function and labels will both be identically zero. In all of our applications, we will set $\mathcal{S}(0) = \infty$ to avoid this trivial solution. In the KPCA case, we forced the variance of the output to be the identity matrix, and so the all-zero solution was not feasible.

3.3 Augmenting Regression with Priors on Functional Form

Suppose that we know f is an element of a Reproducing Kernel Hilbert Space, but we do not know the best form for the kernel function. For example, in the case of a Gaussian kernel,

$$k(\mathbf{x}_1, \mathbf{x}_2) = \exp(-C\|\mathbf{x}_1 - \mathbf{x}_2\|^2) \tag{3.54}$$

modification of the parameter C greatly changes the character of the function space. When C is very small, k is nearly the constant function, and when C is large, $k(\mathbf{x}, \cdot)$ approaches a delta function at $\mathbf{x}$. What is the best way to set this parameter C?

The problem of selecting the kernel best suited to a fitting problem from a family of kernels is called "kernel learning." Kernel learning consists of two components: selecting the best parametrization of a family of kernels and choosing the best cost functional over these parameters. One of the most popular methods used in the machine learning community is minimizing the leave-one-out error [17]. It turns out that even the simple problem of selecting the best parameter C for a Gaussian kernel by minimizing the leave-one-out error is not convex and can be quite computationally intensive even find a local minimum.

Kernel Learning has received a good deal of attention of late. Simple local search methods to directly minimize the Tikhonov cost have been proposed [17], but such minimizations are often not convex and may result in undesirable local minima. Lanckriet et al [56] have shown how to apply semidefinite programming to choose the best convex combination of a finite set of kernels for support vector machines. But if one was searching over more than one parameter, such convex combinations cannot efficiently grid the entire space of kernels. Even for searching for the best convex combination of Gaussian kernels, it has been shown that one kernel for each data point is necessary to achieve optimality [1].

Our approach in this section is to directly minimize the Tikhonov regularization problem augmented with a prior on the output to select the best outputs $\mathbf{y}$ and best

kernel $\mathbf{K}$ in a set $\mathcal{K}$ simaltaneosuly. We have already seen that the arbitrary Tikhonov regularization problem (3.1) is convex in the kernel matrix $\mathbf{K}$ and hence in the kernel function itself. We will show furthermore that as long as we can solve the problem

$$\max_{\mathbf{K}\in\mathcal{K}} \mathbf{v}^\top \mathbf{K}\mathbf{v} \tag{3.55}$$

for all $\mathbf{v} \in \mathbf{R}^N$, then we can solve the dual of the kernel learning problem. In particular, if $\mathcal{K}$ is a convex set, then the resulting primal optimization is convex. When the cost function can be solved by semidefinite program for a fixed $\mathbf{K}$ and if $\mathcal{K}$ can be defined as a projection of the semi-definite cone, then the kernel selection problem can be solved by semidefinite programming. We will end the discussion with three examples of different families of kernels and the computational considerations for each.

3.3.1 The Dual of the Arbitrary Regularization Problem

Whereas in Section 3.1 we derived the representer theorem by dualizing over the RKHS decision variable in the general Tikhonov Regularization problem (3.1), we could have derived a joint dual program over all of the decision variables $\mathbf{z}$, $\mathbf{u}$ and f. In this case we would need to minimize the Lagrangian with respect to all of the variables at once. Furthermore, we can be even more ambitious and dualize over the reproducing kernel itself! Our approach in this section is to directly minimize the general Tikhonov regularization problem (3.1) to select the best outputs $\mathbf{u}$ and best kernel $\mathbf{K}$ in a set $\mathcal{K}$ simultaneously. In particular, we have already seen that if $\mathcal{K}$ is a convex set, then the resulting primal optimization is convex in $\mathbf{K}$. In this regard, all of the problems addressed in this work are convex the kernel matrix $\mathbf{K}$ and hence in the kernel function itself.

Beginning with the optimization (3.9), construct the Lagrangian over all of the variables

$$\mathcal{L}(f, \mathbf{z}, \mathbf{u}, k, \mathbf{c}) = V(\mathbf{z}, \mathbf{u}) + \lambda\langle f, f\rangle_K + 2\lambda \sum_{i=1}^{N} c_i(z_i - \langle f, k(\mathbf{x}_i, \cdot)\rangle_K) \tag{3.56}$$

and minimize over each variable in turn. First, we have seen that for fixed $\mathbf{z}$, $\mathbf{u}$, and k, we can minimize over f to yield

$$V(\mathbf{z}, \mathbf{u}) + 2\lambda \mathbf{c}^\top \mathbf{z} - \lambda \mathbf{c}^\top \mathbf{K}\mathbf{c} \tag{3.57}$$

The remaining optimization is now directly a function of the kernel matrix $\mathbf{K}$. The function

$$V^*(\mathbf{c}, \mathbf{d}) := \max_{\mathbf{z},\mathbf{u}} \mathbf{c}^\top \mathbf{z} + \mathbf{d}^\top \mathbf{u} - V(\mathbf{z}, \mathbf{u}) \tag{3.58}$$

is called the *conjugate dual* of V [12]. It is easy to check that this function is convex. Minimizing with respect to f,$\mathbf{z}$, and $\mathbf{u}$ gives the dual problem

$$\max_{\mathbf{c}} -V^*(-2\lambda \mathbf{c}, 0) - \lambda \max_{\mathbf{K}\in\mathcal{K}} \mathbf{c}^\top \mathbf{K}\mathbf{c} \tag{3.59}$$

There are many consequences of this derivation. First, whereas the primal was a joint optimization over infinite dimensional Hilbert spaces and the functions therein, the dual problem is always an optimization over $\mathbb{R}^N$ where N is the number of data points. In the following section, I will describe a simple subgradient algorithm for optimizing (3.59) when we can efficiently extract the maximizing $\mathbf{K}$ for each $\mathbf{c}$ and can compute V^*.

If V is a strictly convex function and the set $\mathcal{K}$ is convex and has a point in the relative interior, then the optimal value of this optimization is equal to the optimal value of the primal optimization. That is, there is no duality gap. In particular, we can extract the optimal kernel by finding the maximizer

$$\mathbf{K}^* = \arg\max_{\mathbf{K}\in\mathcal{K}} \mathbf{c}^{*\top} \mathbf{K}\mathbf{c}^* \tag{3.60}$$

The specifics of the set $\mathcal{K}$ completely determine the complexity of this kernel learning. For example, if $\mathcal{K}$ can be represented as a set of linear matrix inequalities and V is quadratic or piecewise linear, then (3.59) is a semidefinite program. This generalizes all of the algorithms presented in [56] to generic cost functions with ar-

bitrary priors on the the $\mathbf{u}$ values. In the case that $\mathcal{K}$ is not convex, then the dual of the dual program can be interpreted as searching over the convex hull of $\mathcal{K}$. The final sections highlight some special cases of $\mathcal{K}$.

3.3.2 A Decomposition Algorithm for Solving the Dual Problem and Kernel Learning

We consider a simple subgradient algorithm for solving (3.59) under the assumption that we can solve the problem

$$\max_{\mathbf{K}\in\mathcal{K}} \mathbf{v}^\top \mathbf{K}\mathbf{v} \tag{3.61}$$

and can compute the function $V^*(\mathbf{v})$ for all $\mathbf{v} \in \mathbb{R}^n$.

Define the functions

$$\begin{aligned} q(\mathbf{c}) &:= V^*(2\lambda\mathbf{c}, 0) + m(\mathbf{c}) \\ m(\mathbf{c}) &:= \max_{\mathbf{K}\in\mathcal{K}} \mathbf{c}^\top \mathbf{K}\mathbf{c} \end{aligned} \tag{3.62}$$

and let $\mathcal{D}$ denote the domain of V^*. We seek to minimize q. First, we compute subgradients. Note that

$$\begin{aligned} \partial q(\mathbf{c}) &= 2\lambda \partial V^*(2\lambda\mathbf{c}, 0) + \partial m(\mathbf{c}) \\ \partial V^*(\mathbf{c}) &= \{\mathbf{z}^* \in \mathbb{R}^N \,:\, \mathbf{c}^\top\mathbf{z}^* - V(\mathbf{z},\mathbf{u}) = \max_{\mathbf{z},\mathbf{u}} \mathbf{c}^\top\mathbf{z} - V(\mathbf{z},\mathbf{u}) \\ \partial m(\mathbf{c}) &= \{\mathbf{K}^*\mathbf{c} \,:\, \mathbf{K}^* \in \mathcal{K} \text{ and } \mathbf{c}^\top\mathbf{K}^*\mathbf{c} = \max_{\mathbf{K}\in\mathcal{K}} \mathbf{c}^\top\mathbf{K}\mathbf{c}\} \end{aligned} \tag{3.63}$$

which yields the following subgradient algorithm. Choose a sequence of step sizes t_k with $\sum_{k=1}^{\infty} t_k = \infty$ and $t_k \to 0$.

(i) Begin with a random $\mathbf{c} \in \mathcal{D}$

(ii) Set $\mathbf{s} = \mathbf{z}^* + \mathbf{K}^*\mathbf{c}$ where $\mathbf{z}^* \in \arg\max_{\mathbf{z},\mathbf{u}} 2\lambda\mathbf{c}^\top\mathbf{z} - V(\mathbf{z},\mathbf{u})$ and $\mathbf{K}^* \in \arg\max_{\mathbf{K}\in\mathcal{K}} \mathbf{c}^\top\mathbf{K}\mathbf{c}$.

(iii) Let $\mathbf{c} = (\mathbf{c} - t_k s)_+$ where $(\cdot)_+$ denotes the orthogonal projection onto $\mathcal{D}$

(iv) Let $k = k + 1$ and repeat.

This algorithm will compute an optimal solution, $\mathbf{c}^*$ to the convex program (3.59). Furthermore, when $\mathbf{V}$ is strictly convex the outputs of the optimal function f are given by

$$\begin{aligned} f^*(\mathbf{x}_i) &= z_i \\ \mathbf{z}^* &= \arg\max_{\mathbf{z}} 2\lambda \mathbf{c}^\top \mathbf{z} - V(\mathbf{z}, \mathbf{u}) \end{aligned} \tag{3.64}$$

and if $\mathcal{K}$ is a convex set with a point in its relative interior, then an optimal kernel is a maximizer of $\mathbf{K}^* \in \arg\max_{\mathbf{K}\in\mathcal{K}} \mathbf{c}^\top \mathbf{K}\mathbf{c}$. The analysis of convergence of this algorithm can be found in a variety of places including [7, 11, 12].

3.3.3 Example 1: Finite Set of Kernels

Let us consider the simple case where $\mathcal{K}$ is the finite set of kernels $\{\mathbf{K}_1, \ldots, \mathbf{K}_M\}$. The dual problem is

$$-\min_{\mathbf{c}} (V^*(\mathbf{c}) + \max_{1\le i\le M} \mathbf{c}^\top \mathbf{K}_i \mathbf{c}) \tag{3.65}$$

or

$$\begin{aligned} -\min_{\mathbf{c},\gamma} \quad & (V^*(\mathbf{c}) + \gamma) \\ \text{s.t.} \quad & \mathbf{c}^\top \mathbf{K}_i \mathbf{c} \le \gamma \end{aligned} \tag{3.66}$$

the associated dual of this problem amounts to learning the best convex combination of the M kernels

$$\begin{aligned} \min_{\mathbf{z},t,\mathbf{p}} \quad & V(\mathbf{z}, \mathbf{y}) + \lambda t \\ \text{s.t.} \quad & \begin{bmatrix} t & \mathbf{z}^\top \\ \mathbf{z} & \sum_{i=1}^M p_i \mathbf{K}_i \end{bmatrix} \succeq 0 \\ & \mathbf{p} \ge 0, \quad \sum_{i=1}^M p_i = 1 \end{aligned} \tag{3.67}$$

In particular, this semidefinite program generalizes the algorithm in [56] where the cost function was assumed to be a support vector machine loss. This problem takes

on an interesting form when the cost is the least-squares cost

$$\min_{f,\mathbf{K}} \sum_{i=1}^{L} (f(\mathbf{x}_i) - y_i)^2 + \lambda \|f\|^2 \tag{3.68}$$

As we have noted before, applying the Representer Theorem and solving for f gives the Tikhonov cost

$$\mathbf{C}_T(\mathbf{K}) := \lambda \mathbf{y}^\top (\mathbf{K} + \lambda \mathbb{1})^{-1} \mathbf{y} \tag{3.69}$$

This is convex in $\mathbf{K}$. Indeed if we parameterize $\mathbf{K}$ as

$$\mathbf{K} = \sum_{i=1}^{T} \alpha_i \mathbf{K}_i \tag{3.70}$$

then by the Schur Complement Lemma

$$\begin{aligned} \min_{\alpha} \lambda \mathbf{y}^\top (\mathbf{K} + \lambda \mathbb{1})^{-1} \mathbf{y} = \min_{t,\alpha} \quad & \lambda t \\ \text{s.t.} \quad & \begin{bmatrix} t & \mathbf{y}^\top \\ \mathbf{y} & \sum_{i=1}^{T} \alpha_i \mathbf{K}_i + \lambda \mathbb{1} \end{bmatrix} \succeq 0 \end{aligned} \tag{3.71}$$

which is a semidefinite program.

We may apply any set of linear constraints to the α_i and preserve convexity. In particular, if α_i are on the T-simplex then we are searching for the best convex combination of a given set of kernels.

$$\begin{aligned} \min_{t,\alpha} \quad & \lambda t \\ \text{s.t.} \quad & \begin{bmatrix} t & \mathbf{y}^\top \\ \mathbf{y} & \sum_{i=1}^{T} \alpha_i \mathbf{K}_i + \lambda \mathbb{1} \end{bmatrix} \succeq 0 \\ & \alpha_i \geq 0 \quad \sum_{i=1}^{T} \alpha_i = 1 \end{aligned} \tag{3.72}$$

Since $C(z) = (y - z)^2$, then we have seen in Section 3.2.1 that $C*(c) = \frac{1}{4}c^2 + cy$.

The dual program for this problem is thus given by

$$\begin{array}{ll} \max_{\mathbf{c},\gamma} & -\mathbf{c}^\top \mathbf{c} + 2\mathbf{y}^\top \mathbf{c} - \gamma \\ \text{s.t.} & \mathbf{c}^\top \mathbf{K}_i \mathbf{c} \geq \lambda\gamma \end{array} \tag{3.73}$$

This is quite similar to Lanckriet's Kernel selection program for the SVM [56]. In particular, this can be solved using a second-order cone programming solver [61] which is generally more efficient than solving a semidefinite program.

3.3.4 Example 2: Gaussian Kernels

Let $\mathcal{K}$ be the set of all Gaussian kernels

$$\mathcal{K} = \{\mathbf{K}(\mathbf{x},\mathbf{y}) = \exp(-\theta\|\mathbf{x}-\mathbf{y}\|^2) \,:\, \theta > 0\} \tag{3.74}$$

For a fixed data set, let $\mathbf{K}[\theta]$ denote the kernel matrix generated from the Gaussian kernel with parameter θ. In this case the dual of the kernel selection problem is the semi-infinite program

$$\begin{array}{ll} -\min\limits_{\mathbf{c},\gamma} & (V^*(\mathbf{c}) + \gamma) \\ \text{s.t.} & \mathbf{c}^\top \mathbf{K}[\theta]\mathbf{c} \leq \gamma \quad \forall\theta \geq 0 \end{array} \tag{3.75}$$

This problem is convex and strictly feasible as if we let $\gamma \geq \sup_\theta \mathbf{c}^\top \mathbf{K}[\theta]\mathbf{c}$, we can satisfy the all of the inequalities strictly.

The dual of this dual problem amounts to selecting the best kernel from the convex hull of the Gaussian kernels. Using the dual, we can provide an elementary proof that the optimal selection for convex combination of Gaussian kernels for a data set with N points is given by a convex combination of N kernels. This result was derived in [1] by a lengthy argument assuming a differentiable cost V.

However, for an arbitrary convex cost, the expansion is an immediate result of classic results from semi-infinite optimization (see, for example, [36, 44, 62] for a proof).

Theorem 3.3.1 *Given a semi-infinite optimization*

$$\begin{aligned} \min_{\mathbf{x}} \quad & f(\mathbf{x}) \\ s.t. \quad & g(\mathbf{x},\mathbf{y}) \leq 0, \quad ,\mathbf{y} \in \Omega \end{aligned} \tag{3.76}$$

with $\mathbf{x} \in \mathbb{R}^n$, $\Omega \subset \mathbb{R}^m$ *compact,* f *convex, suppose there exists a* $\hat{\mathbf{x}}$ *such that* $g(\hat{\mathbf{x}},\mathbf{y}) < 0$ *for all* $\mathbf{y} \in \Omega$. *Then* $\mathbf{x}^*$ *is a global minimizer of (3.76) if and only if there exist* $\mu_1,\ldots,\mu_n \geq 0$, $\mathbf{y}_1,\ldots,\mathbf{y}_n \in \Omega$ *such that*

$$\nabla f(\mathbf{x}^*) + \sum_{i=1}^{n} \mu_i \nabla g(\mathbf{x}^*,\mathbf{y}_i) = 0 \tag{3.77}$$

The proof of this theorem is a mild generalization of the classical Slater condition to the semi-infinite case. One shows that 0 is in the convex hull of $\{\nabla f(\mathbf{x}^*)\} \cup \{\nabla g(\mathbf{x}^*,\mathbf{y}) : \mathbf{y} \in \Omega\}$ and that the coefficient of $\nabla f(\mathbf{x}^*)$ can always be chosen to be non-zero. Then, by Caratheodory's theorem, such a convex combination can be found using only n vectors.

As an immediate consequence, we have

Corollary 3.3.2 *For any convex cost* V, *the optimal kernel for (3.86) with* $\mathcal{K}$ *as in (3.74), the optimal* $\mathbf{K}$ *is given by*

$$\mathbf{K}^* = \sum_{i=1}^{N} p_i \mathbf{K}[\theta_i] \tag{3.78}$$

for some nonnegative p_i,θ_i *with* $\sum_i p_i = 1$

Proof The optimization (3.75) is strictly feasible. By Theorem 3.3.1, a necessary and sufficient condition for optimality of is the existence of p_i,θ_i with $\sum_i p_i = 1$ such that

$$\mathbf{z}^* + (\sum_{i=1}^{N} p_i \mathbf{K}[\theta_i])\mathbf{c}^* = 0 \tag{3.79}$$

and that there is no duality gap. Thus, the optimal kernel has the desired form. Equation (3.79) is nothing more than the KKT conditions for the dual problem. ■

From a practical perspective, this result is discouraging. It seems excessive to have one kernel for every data point in the training set. This is especially true in the case of the Gaussian kernels where there is only one parameter to optimize. For smooth cost functions V, the cost function (3.86)is Lipshitz continuous in the single parameter C, and, arguably, a simple search like Brent's 1-d optimization algorithm would find the optimal solution by solving the Tikhonov regularization problem for a few choices of C.

Furthermore, even when content to search for a large convex combination of kernels, the optimal centers θ_i can be difficult to find. Argyriou et al [1] propose a greedy algorithm that is not even guaranteed to converge after n steps. Indeed, it may converge arbitrarily slowly. For these reasons, it is interesting to consider the case of *polynomials* where the search is convex in the kernel parameters.

3.3.5 Example 3: Polynomial Kernels

Polynomials are attractive for kernel learning as the positivity of polynomial kernels is easy to characterize and they can be parametrized as a subset of the semidefinite cone. In this section, we will first characterize the set of polynomial functions that are positive definite. Then we will describe semidefinite program for selecting the optimal polynomial kernel.

Let $\mathbf{x} \in \mathbb{R}^d$ and denote the vector of monomials of degree less than or equal to p by $\tilde{\mathbf{x}}_p$.

Theorem 3.3.3 *Let $k(\mathbf{x}, \mathbf{y})$ be a symmetric polynomial of degree $2p$ on a compact set of infinite cardinality $\Omega \subset \mathbb{R}^d$. Then $k(\mathbf{x}, \mathbf{y})$ is a positive definite kernel on Ω if and only if*

$$k(\mathbf{x}, \mathbf{y}) = \tilde{\mathbf{x}}_p^\top \mathbf{Q} \tilde{\mathbf{y}}_p \tag{3.80}$$

for some positive semidefinite matrix $\mathbf{Q}$.

There are several properties of this theorem that are worth remarking upon. First, for those familiar with positive polynomials, note that there is no ambiguity in the

definition of the matrix $\mathbf{Q}$ unlike in the case of polynomials where $\mathbf{Q}$ is only defined as an affine subspace. Second, in the kernel learning optimization (3.86), the matrix $\mathbf{Q}$ appears linearly in the constraint and we can directly minimize with respect to this variable. Any prior information about the terms to include in $\mathbf{Q}$ or relations amongst the components of $\mathbf{Q}$ can be immediately included in the optimization. We now present a simple proof for the theorem.

Proof [of Theorem 3.3.3]

Sufficiency of Equation (3.80) can be seen by picking $\mathbf{x}_1, \ldots, \mathbf{x}_N \in \Omega$ and $c_1, \ldots, c_N \in \mathbb{R}$. Let $\tilde{\mathbf{X}} = [\mathbf{x}_1, \ldots, \mathbf{x}_N]$ and $\mathbf{c} = [c_1, \ldots, c_N]^\top$. Then since $\mathbf{Q} \succeq 0$ we have

$$\sum_{i,j=1}^{N} c_i c_j k(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{c}^\top \tilde{\mathbf{X}}^\top \mathbf{Q} \tilde{\mathbf{X}} \mathbf{c} \geq 0 \tag{3.81}$$

proving that k is a positive definite kernel.

To prove the converse, first note that by Mercer's Theorem,

$$k(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^{\infty} \lambda_i \Phi_i(\mathbf{x}) \Phi_i(\mathbf{y}) \tag{3.82}$$

where

$$\int_\Omega k(\mathbf{x}, \mathbf{y}) \Phi_i(\mathbf{y}) d\mathbf{y} = \lambda_i \Phi_i(\mathbf{x}) \tag{3.83}$$

and $\lambda_i \geq 0$. The theorem will be proven if we can show that the series (3.82) is finite and that all of the eigenfunctions are polynomials. In this case, we can construct the positive semidefinite matrix $\mathbf{Q}$ in Equation (3.80) by letting $\mathbf{q}_i$ be the vector such that $\Phi_i(\mathbf{x}) = \mathbf{q}_i^\top \tilde{\mathbf{x}}_p$. Plugging this into the Mercer expansion gives

$$k(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^{I} \lambda_i \tilde{\mathbf{x}}^\top \mathbf{q}_i \mathbf{q}_i^\top \tilde{\mathbf{y}} \tag{3.84}$$

Let $\mathbf{Q} := \sum_{i=1}^{I} \lambda_i \mathbf{q}_i \mathbf{q}_i^\top$. It is a positive combination of outer products and hence must be positive semidefinite as desired.

To prove that the Mercer expansion is a finite sum of polynomials, consider the

functions indexed by $\mathbf{y} \in \Omega$

$$f_{\mathbf{y}}(\mathbf{x}) := k(\mathbf{y}, \mathbf{x}) \tag{3.85}$$

Finite sums $\sum_{j=1}^{N} \alpha_j f_{\mathbf{y}_j}$ are polynomials of degree at most p. Hence, the span of the $f_{\mathbf{y}}$ lies in the finite dimensional subspace of $\mathcal{L}_2(\Omega)$ consisting of polynomials of degree less that or equal to p. Since all finite dimensional subspaces of $\mathcal{L}_2(\Omega)$ are closed, all of the limit points of the set of $f_{\mathbf{y}}$ must be polynomials of degree less that or equal to p. In particular, the integral on the left hand side of (3.83) is a limit point of the $f_{\mathbf{y}}$ and must be a polynomial. Consequently all of the Φ_i are polynomials of degree at most p. It remains to prove that there are only a finite number of Φ_i in the expansion. But this follows because Φ_i are an orthonormal sequence in a finite dimensional subspace of $\mathcal{L}_2(\Omega)$, completing the proof. ■

Algorithmically, we can take advantage of this theorem as follows. First, let $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_N]$ denote the data matrix and $\tilde{\mathbf{X}} = [\tilde{\mathbf{x}}_1, \ldots, \tilde{\mathbf{x}}_N]$ denote the matrix of data lifted to a list of monomials of degree less than or equl to p. The kernel learning optimization for polynomial kernels is given by

$$\begin{aligned} \min_{\mathbf{z},t,\mathbf{Q}} \quad & V(\mathbf{z}, \mathbf{y}) + \lambda t \\ \text{s.t.} \quad & \begin{bmatrix} t & \mathbf{z}^\top \\ \mathbf{z} & \tilde{\mathbf{X}}^\top \mathbf{Q} \tilde{\mathbf{X}} \end{bmatrix} \succeq 0 \\ & \mathbf{Q} \succeq 0 \end{aligned} \tag{3.86}$$

By making $\mathbf{Q}$ sufficiently large, this cost of this optimization can be made arbitrarily small. Thus, we must constrain the maximal value for $\mathbf{Q}$. This can be done in a variety of ways, but the easiest constraint to adjoin is $\mathrm{Tr}(\mathbf{Q}) \leq \beta$ for some $\beta > 0$. This can be also adjoined to the cost as a penalty function when the value β is not known explicitly. On the other hand, there is no reason not to just have $\beta = 1$ as this simply sets a scale for the polynomial feature space.

3.4 Conclusion

In this chapter, I have shown that an arbitrary Tihkonov cost function can be augmented with priors on hidden labels, and then jointly optimized over the functions, kernels, and labels. Using tools from Lagrangian duality, I have shown that these optimizations are tractable, and recover a variety of learning algorithms.

The remainder of this document will focus on particular instances of the cost function $\mathcal{S}(\mathbf{y})$. In Chapter 4, I will require that $y_i = \pm 1$. This problem, though NP-HARD, will generalize the standard kernel methods for clustering and transduction. In Chapter 5, I will require that $\mathbf{y}$ lie near the outputs of a linear dynamical system. This will serve as a powerful method for transforming time series with very few examples and dimensionality reduction. Finally, in Chapter 6, I will require that $\mathbf{y}$ be a constant vector. This will result in a method for learning manifolds of low-codimension with applications in anomaly detection. As we have seen, all of these algorithms are also convex in the kernel function, so if prior information on functional form is provided, then it can be incorporated into these learning algorithms as well.

Chapter 4

Output Prior: Binary Labels

One of the most successful applications of Tikhonov regularization on Reproducing Kernel Hilbert Spaces has been data driven classifiers. Classifiers return discrete class labels as output. For instance, a classifier could take as input an image and return whether the image was of a cat or a dog, or it might take a piece of music and return whether it was written by Mozart or Beethoven. A common approach to training such classifiers is to provide labels $y = \pm 1$ for each $\mathbf{x}$ in the training set. If a training example is in the first class it is assigned a 1. If it is in the second class, it is assigned a -1.

The classification algorithms differ in their choice of cost function. Most famous is Vapnik's Support Vector Machine [13, 101] where the cost function is the so-called *hinge loss*

$$V(f(\mathbf{x}), y) = \max(0, 1 - f(\mathbf{x})y)\,. \tag{4.1}$$

This cost function assigns no penalty if $f(\mathbf{x})$ is of the same sign as y and is of greater magnitude than 1 and assigns a linearly increasing penalty otherwise. Another method, regularized least-squares classification, just uses the least-squares cost. Even though it might not make intuitive sense to use a least squares cost for classification, the RLSC algorithm, when well tuned, performs as well as the SVM and has certain computational advantages [84, 83].

In this chapter we will look at the situation where some of the class labels are

withheld. In the absence of the labels, we will constrain the output values to be either $+1$ or -1. This constraint is non-convex, and as we will see, finding the optimal labels is NP-HARD. However, we will show that we can approximate the optimal labels with a variety of algorithms. Using Lagrangian duality, we will develop a class of semidefinite programming problems that approximate the semi-supervised classification and segmentation problems arising from Tikhonov regularization with quadratic or polyhedral cost functions.

The semidefinite programs we derive can be solved quickly for sets of a few hundred examples. For problems where there are thousands of variables, however, the semidefinite programming problems may not even fit in memory. To fix this problem, we discuss a class of eigenvalue approximations derived from these semidefinite programs. These approximations include the well-known *spectral clustering* algorithms [103]. In particular, this derivation reveals the functions that the spectral clustering algorithms are implicitly learning.

The semi-supervised classification problem is often called *transduction.* Several algorithms have been proposed to solve the transductive problem introduced in [100]. Joachims provides a local search method for the transductive SVM which is fast but subject to local minima [49]. Similarly, Smola et al present a local-search method for solving the transductive problem for the least-squares cost [94]. In [25], the authors construct a relaxation of the standard support vector machine problem by relaxing the rank one product outer product of the labels vector. Unfortunately, since they relax the dual form of the SVM rather than the primal Tikhonov regularization problem, the resulting semidefinite program cannot be solved for more than one hundred variables. The relaxation we present can be solved for a several thousand variables by solving the associated dual.

In the case when no labels are given at all, we are searching for a function which divides a data set into two classes. This unsupervised learning can be interpreted as *segmentation* or *clustering*, where, instead of fitting a mixture model to tessellate the data, we search for a smooth function whose zero set passes through the data separating it into two sets.

The Normalized Cuts clustering algorithm of Shi and Malik, although originally presented as spectral relaxation of a graph-cut problem, can be interpreted as a relaxation of the unsupervised Tikhonov regularization problem. Normalized Cuts views the data set as a graph, where nodes represent data points and edges are weighted according to the similarity, or "affinity", between data points. This is the starting point of many other graph-based clustering algorithms [103, 2]. The affinity matrix used in these algorithms is a kernel matrix derived from some positive definite function and hence is a Gram matrix on some RKHS. We show that Normalized Cuts may be interpreted as learning a function in this RKHS that labels points by the sign of this function.

This new interpretation of Normalized Cuts reveals that it weights data points away from the mean of the data set more than those in the center of the data set. This weighting causes Normalized Cuts to sometimes break elongated clusters and to be sensitive to outliers. By defining a eigenvalue relaxation that gives equal weight to all data points, we derive a clustering algorithm (the Average Gap algorithm) that does not exhibit these problems. Finding labels under this new gap reduces to thresholding the top eigenvector of a matrix.

This chapter only presents 2-way clustering algorithms. If more clusters are sought, each 2-way cut can be further subdivided by running the clustering procedure recursively [90].

4.1 Transduction, Clustering, and Segmentation via constrained outputs

Since most classification algorithms are trained with y labels set to either $+1$ or -1, an intuitive solution to utilize the unlabelled data is to constrain the unlabelled data to be either plus one or minus one. Let $L = N + M$ with the points $\mathbf{x}_1, \ldots, \mathbf{x}_N$ labelled the points $\mathbf{x}_{N+1}, \ldots, \mathbf{x}_{N+M}$ unlabelled. The optimization we shall study for the remainder of the chapter is

Problem 1 RKHS Transduction *Find a set of assignments to optimize*

$$\begin{aligned} \min_{f,\mathbf{u}_M} \quad & \sum_{i=1}^{L} C(f(\mathbf{x}_i), u_i) + \lambda \|f\|_K^2 \\ s.t. \quad & u_i = y_i \quad i = 1, \ldots, N \\ & u_i \in \{-1, 1\} \quad i = N+1, \ldots, N+M \end{aligned} \tag{4.2}$$

The resulting algorithm depends only on the choice of the cost function C and how to best constrain u_i to be binary. This optimization seeks to minimize a convex function over the non-convex set $\mathbf{u} \in \mathbb{R}^M : u_i = \pm 1$. We will see in the next section that even for the least squares cost, this problem is NP-HARD for a generic RKHS.

There are several different cost functions one could consider. The first is the least-squares cost

Problem 2 Transductive Regularized Least Squares *Find a minimizer for (4.2) when* $C(f(\mathbf{x}), y)) = (f(\mathbf{x}) - y)^2$

As previously derived, we may plug in $C(f(\mathbf{x}), y) = (f(\mathbf{x}) - y)^2$ and solve for the optimal f to yield the optimization over the unlabelled points

$$\begin{aligned} \min_{\mathbf{u}_M} \quad & \lambda \begin{bmatrix} \mathbf{y}_N \\ \mathbf{u}_M \end{bmatrix}^\top (\mathbf{K} + \lambda \mathbb{1})^{-1} \begin{bmatrix} \mathbf{y}_N \\ \mathbf{u}_M \end{bmatrix} \\ & u_i \in \{-1, 1\} \quad i = N+1, \ldots, N+M \end{aligned} \tag{4.3}$$

There are two different costs which we will consider, inspired by the support vector machine. The first is the hinge loss. The second is a hard margin loss that does not allow for $y_i f(\mathbf{x})$ to be less than 1.

Problem 3 Transductive Support Vector Machine *Find a minimizer for (4.2) when* $C(f(\mathbf{x}), y)) = \max(0, 1 - f(\mathbf{x})y)$

Problem 4 Hard Margin Transductive Support Vector Machine *Find a minimizer for (4.2) when* $C(f(\mathbf{x}), y)) = \delta_+(1 - f(\mathbf{x})y)$

For both notational simplicity and the clarity of presentation, we will formulate the SVM transduction problems using the form of (3.15) in Chapter 3:

$$\begin{aligned} \min_{\mathbf{z},t} \quad & V(\{(z_i, y_i)\}) + \lambda t \\ \text{s.t.} \quad & \begin{bmatrix} t & \mathbf{z}^\top \\ \mathbf{z} & \mathbf{K} \end{bmatrix} \succeq 0 \end{aligned} \tag{4.4}$$

Let the matrix $\mathbf{A}$ span the null-space of the kernel $\mathbf{K}$. Then, by applying the Schur complement lemma and Lemma A.7.2 from Chapter 3 we can rewrite (4.4) as

$$\begin{aligned} \min_{\mathbf{z}} \quad & V(\{(z_i, y_i)\}) + \lambda \mathbf{z}^\top \mathbf{K}^\dagger \mathbf{z} \\ \text{s.t.} \quad & \mathbf{A}\mathbf{z} = 0 \end{aligned} \tag{4.5}$$

Let us apply this to Problems 3 and 4. First, the transduction problem with the standard form of the SVM with a hinge loss is given by

$$\begin{aligned} \min_{\mathbf{z},\mathbf{u}_M} \quad & \sum_{i=1}^{N+M} \xi_i + \lambda \mathbf{z}\mathbf{K}^\dagger\mathbf{z} \\ \text{s.t.} \quad & u_i z_i \geq 1 - \xi_i \\ & u_i = y_i \quad i = 1, \ldots, N \\ & u_i \in \{-1, 1\} \quad i = N+1, \ldots, N+M \\ & \xi_i \geq 0 \\ & \mathbf{A}\mathbf{z} = 0 \end{aligned} \tag{4.6}$$

Second, the using the hard margin, we get the problem

$$\begin{aligned} \min_{\mathbf{z},\mathbf{u}_M} \quad & \lambda \mathbf{z}\mathbf{K}^\dagger\mathbf{z} \\ \text{s.t.} \quad & u_i z_i \geq 1 \\ & u_i = y_i \quad i = 1, \ldots, N \\ & u_i \in \{-1, 1\} \quad i = N+1, \ldots, N+M \\ & \mathbf{A}\mathbf{z} = 0 \end{aligned} \tag{4.7}$$

In this case, the regularization parameter λ is only acting to scale the cost function, and does not directly effect the optimal $\mathbf{z}$ or $\mathbf{u}_M$. We will omit this parameter in the further references to the this problem. We will show in the next section that all three cost functions are NP-HARD.

Before proceeding, let us remark that in the case that no labelled examples are available, we need to ensure that the trivial labelling $f(\mathbf{x}_i) = 1$ is not in the RKHS. When there are no labels, we will adjoin the constraint $\sum_{i=1}^{N} f(\mathbf{x}_i) = 0$ to the optimization to avoid this redundancy.

4.2 RKHS Clustering is NP-HARD

We will first show that the clustering problem with least squares cost and no labels, is NP-HARD. Then we will show that providing an incomplete set of labels the semisupervised problem is also NP-HARD. Finally, we will show that the variants related to the SVM are also NP-HARD.

While the following arguments should the reader that is unreasonable to expect to solve these segmentation problems exactly, this section should not be discouraging! Even the well-known problem of K-Means clustering is NP-HARD, yet heuristic coordinate ascent algorithms have proven successful for many applications. The remainder of the chapter will apply the approximation techniques from Chapter 2 to yield powerful algorithms that, in practice, perform quite well.

Our proofs of hardness will be through reductions from the Number Partitioning Problem, one of the fundamental NP-Complete problems [32].

Problem 5 The number partitioning problem (NPP): *Given a set of integers $A = \{a_1, \ldots, a_N\}$, does there exist a partition of A into two subsets U and V such that the respective subset sums are equal?*

This problem is NP-complete even under particular restrictions. For example, if one requires that $||U| - |V|| \leq 1$, the problem is NP-HARD. Similarly, if, for a fixed constant c, we are given the assignments for the first cN elements, the problem is NP-HARD.

Problem 6 The "semi-supervised" number partitioning problem (SNPP): *Given a set of integers* $A = \{a_1, \ldots, a_L\}$, *two integers* l *and* k *such that* $k + l < cL$, *does there exist a partition of* A *into two subsets* U *and* V *such* $a_1, \ldots, a_k \in U$, $a_{k+1}, \ldots, a_{k+l} \in V$ *such that that the respective sums are equal?*

This problem is equivalent to the Problem 5. To see this define a new set

$$B = \{\sum_{i=1}^{k} a_i - \sum_{j=k+1}^{l} a_j, a_{l+1}, a_{l+2}, \ldots, L\} \tag{4.8}$$

If the answer to Problem 6 is YES, then let U and V be the partitions of the set A. Define U' to be the set of all elements $a_j \in U$ with $j > k + l$. Define V' to be the set of all elements $a_j \in V$ with $j > k + 1$. Then $U' \cup \{\sum_{i=1}^{k} a_i - \sum_{j=k+1}^{l} a_j\}$ and V' are partitions of B with equal subset sums. Similarly, if there is a partition of B into two sets with equal subset sums, we can construct a partition for A. As a result of this discussion, we will show in what follows that the unsupervised learning problem is generically hard, and, as an immediate corollary we will deduce that the semi-supervised problem is also NP-HARD.

The following lemma identifies NPP with a particular quadratic optimization. We will reduce the optimization to this form by picking a particular set of input data.

Lemma 4.2.1 *The number partitioning problem is equivalent to finding a minimizer of the binary quadratic program*

$$\begin{aligned} \min_{\mathbf{u}} \quad & \mathbf{u}^\top \mathbf{a}\mathbf{a}^\top \mathbf{u} \\ s.t. \quad & u_i^2 = 1 \end{aligned} \tag{4.9}$$

Proof Let U and V partition A. Let $u_i = +1$ if $a_i \in U$ and $u_i = -1$ if $a_i \in V$. Then the difference between the subset-sums of U and V is

$$\sum_{a_j \in U} a_j - \sum_{a_k \in V} a_k = \sum_{i=1}^{L} u_i a_i \tag{4.10}$$

Therefore, A can be partitioned into a pair of sets with equal sums if and only if there

exists a y such that $\sum_{i=1}^{L} u_i a_i = 0$. Since

$$\mathbf{u}^\top \mathbf{a}\mathbf{a}^\top \mathbf{u} = (\sum_{i=1}^{L} u_i a_i)^2 \geq 0\,, \tag{4.11}$$

then A has a partition if and only if the minimum of (4.9) is zero. ■

The following theorem is an immediate consequence of the preceding discussion.

Theorem 4.2.2 *Problem 2 is NP-HARD.*

Proof Given an instance of NPP, we will reduce it to a clustering problem as follows. Let $\mathbf{a} = [a_1, \ldots, a_n]^\top$ be the column vector containing the elements of the set $\mathbf{a}$. For $i = 2, \ldots, N$, Let $\{\mathbf{a}/||\mathbf{a}||, \mathbf{v}_2, \ldots, \mathbf{v}_N\}$ be an orthonormal basis for $\mathbf{R}^n$. Let

$$\mathbf{K} = \begin{bmatrix} \mathbf{v}_2 & \ldots & \mathbf{v}_N \end{bmatrix} \begin{bmatrix} \mathbf{v}_2 & \ldots & \mathbf{v}_N \end{bmatrix}^\top \tag{4.12}$$

Finally, let λ be any positive rational number. It is easy to check that

$$(\frac{1}{\lambda}\mathbf{K} + \mathbb{1})^{-1} = \frac{1}{(1+\lambda)||\mathbf{a}||^2}\mathbf{a}\mathbf{a}^\top + \frac{\lambda}{1+\lambda}\mathbb{1} \tag{4.13}$$

and hence

$$\arg\min_{u_i^2=1} \mathbf{u}^\top (\frac{1}{\lambda}\mathbf{K} + \mathbb{1})^{-1}\mathbf{u} = \arg\min_{u_i^2=1} \mathbf{u}^\top \mathbf{a}\mathbf{a}^\top \mathbf{u} \tag{4.14}$$

therefore, if one can find the optimal cluster assignments, one can find the optimal partitioning of A. It is clear that the analysis remains the same if some of the y_i labels are given or if we require the two clusters to be balanced. This completes the proof. ■

Since Problem 2 is a subproblem of Problem 1, we have the following

Corollary 4.2.3 *Problem 1 is NP-HARD.*

To proceed to the proofs that transduction with the Support Vector Machine costs is NP-HARD, we introduce a variant of the quadratic program for least-squares clustering with *inequality* constraints

Problem 7 *Given a positive definite $N \times N$ matrix $\mathbf{Q}$, find a minimizer of the optimization*

$$\begin{aligned} \min_{\mathbf{u}} \quad & \mathbf{u}^\top \mathbf{Q} \mathbf{u} \\ s.t \quad & u_i^2 \geq 1 \end{aligned} \tag{4.15}$$

Theorem 4.2.4 *Problem 7 is NP-HARD*

Proof We will show how this problem can be used to solve the NPP problem. Let $\mathbf{a}$ be an integer vector. There exists a partition of this vector into two subsets with equal sums if we can find a binary $\mathbf{u}$ such that $\mathbf{u}^\top \mathbf{a}\mathbf{a}^\top \mathbf{u} < 1$.

Let $\mathbf{Q}_\beta = \mathbf{a}\mathbf{a}^\top + \beta \mathbb{1}$. Let

$$\mathbf{u}_\beta = \arg\min_{u_i^2 \geq 1} \mathbf{u}^\top \mathbf{Q}_\beta \mathbf{u} \tag{4.16}$$

then for all $\mathbf{q}$ with $q_i \in \{+1, -1\}$,

$$\mathbf{u}_\beta^\top \mathbf{a}\mathbf{a}^\top \mathbf{u}_\beta + \beta \|\mathbf{u}_\beta\|^2 \leq \mathbf{q}^\top \mathbf{a}\mathbf{a}^\top \mathbf{q} + \beta \|\mathbf{q}\|^2 \tag{4.17}$$

Since $q_i^2 = 1$ for all i, $\|\mathbf{q}\|^2 = N$, so we have

$$\beta(\|\mathbf{u}_\beta\|^2 - N) \leq \mathbf{q}^\top \mathbf{a}\mathbf{a}^\top \mathbf{q} - \mathbf{u}_\beta^\top \mathbf{a}\mathbf{a}^\top \mathbf{u}_\beta \leq \mathbf{q}^\top \mathbf{a}\mathbf{a}^\top \mathbf{q} \leq \max_{z_i \in \{-1,+1\}} \mathbf{q}^\top \mathbf{a}\mathbf{a}^\top \mathbf{q} = \|\mathbf{a}\|_1 \tag{4.18}$$

Since the left-hand side is bounded for all β, we have

$$\lim_{\beta \to \infty} \|\mathbf{u}_\beta\|^2 = N \tag{4.19}$$

Furthermore, it is easy to check that for $\mathbf{u}$ with all entries of magnitude greater than or equal to 1, the closest vector of norm N is $\mathbf{v} = \text{sign}(\mathbf{u})$.

Since the function $\mathbf{u}^\top \mathbf{a}\mathbf{a}^\top \mathbf{u}$ is continuous, there exists a δ such that if $\| \text{sign}(\mathbf{u}_\beta) - \mathbf{u}_\beta \| < \delta$, then

$$|\text{sign}(\mathbf{u}_\beta)^\top \mathbf{a}\mathbf{a}^\top \text{sign}(\mathbf{u}_\beta) - \mathbf{u}_\beta^\top \mathbf{a}\mathbf{a}^\top \mathbf{u}_\beta| < \frac{1}{2} \tag{4.20}$$

if we choose β large enough, then $\| \operatorname{sign}(\mathbf{u}_\beta) - \mathbf{u}_\beta \| < \delta$ and hence we can solve the problem NPP. ■

Now, as corollaries, we can show

Theorem 4.2.5 *Problem 4 is NP-HARD.*

Proof Let $\mathbf{u} = \mathbf{K}\mathbf{c}$. Then we can rewrite Problem 4 as

$$\begin{aligned} \min_{\mathbf{u}} \quad & \mathbf{u}^\top \mathbf{K}^\dagger \mathbf{u} \\ \text{s.t.} \quad & u_i \geq 1 \\ & \mathbf{1}^\top \mathbf{u} = 0 \\ & \mathbf{A}\mathbf{u} = 0 \end{aligned} \tag{4.21}$$

of which Problem 7 is a special case. ■

Theorem 4.2.6 *Problem 3 is NP-HARD.*

Proof By making the regularization parameter small, we can approximate the solution to Problem 4. In particular, this would lead to being able solve NPP. ■

4.3 Semidefinite Approximation using Lagrangian Duality

As noted in the previous section and in Chapter 2, we may identify the constraint $u_i \in \{-1, 1\}$ with the quadratic identity $u_i^2 = 1$. By using this constraint, we may consider all subproblems of Problem 1 with quadratic or polyhedral costs V as Non-convex Quadratically Constrained Quadratic Programs (NCQCQP). This set of optimization problems are, as discussed in Chapter 2, NP-HARD, but admit powerful approximation algorithms via their associated Lagrangian dual problem.

Here we present the primal-dual pair of semidefinite programs associated with the RLS cost. We will use this cost function and the random hyperplane algorithm presented in Chapter 2 to produce feasible labels assignments for the unlabelled data.

Given $N+M$ data points $\mathbf{x}_1, \ldots, \mathbf{x}_{N+M}$ and labels for the first N points, $y_1, \ldots, y_N$, we want to infer cluster assignments for the remaining M points. Let us group as vectors $\mathbf{y}_N$ the labels of the first N data points and $\mathbf{u}_M$ the (unknown) labels of the last M data points. Also, let us partition the Kernel matrix of the data as

$$\mathbf{K} = \begin{bmatrix} \mathbf{K}_{NN} & \mathbf{K}_{NM} \\ \mathbf{K}_{MN} & \mathbf{K}_{MM} \end{bmatrix} \tag{4.22}$$

Let $\mathbf{Q} = (\mathbf{K} + \lambda \mathbb{1})^{-1}$ and partition this matrix like $\mathbf{K}$ to give the optimization

$$\begin{aligned} \min_{\mathbf{u}_M} \quad & \begin{bmatrix} \mathbf{y}_N \\ \mathbf{u}_M \end{bmatrix}^\top \begin{bmatrix} \mathbf{Q}_{NN} & \mathbf{Q}_{NM} \\ \mathbf{Q}_{MN} & \mathbf{Q}_{MM} \end{bmatrix} \begin{bmatrix} \mathbf{y}_N \\ \mathbf{u}_M \end{bmatrix} \\ \text{s.t.} \quad & u_i^2 = 1 \text{ for } i = N+1, \ldots, N+M \end{aligned} \tag{4.23}$$

This may be further simplified by lumping together the labelled data points. Let $\mathbf{y}_p := \mathbf{Q}_{MN}\mathbf{y}_N$ and let $T := \mathbf{y}_N^\top \mathbf{Q}_{NN}\mathbf{y}_N$. Then the optimization simplifies to

$$\begin{aligned} \min_{z,\mathbf{u}_M} \quad & \begin{bmatrix} z \\ \mathbf{u}_M \end{bmatrix}^\top \begin{bmatrix} T & \mathbf{y}_p^\top \\ \mathbf{y}_p & \mathbf{Q}_{MM} \end{bmatrix} \begin{bmatrix} z \\ \mathbf{u}_M \end{bmatrix} \\ \text{s.t.} \quad & u_i^2 = 1 \text{ for } i = N+1, \ldots, N+M \\ & z^2 = 1 \end{aligned} \tag{4.24}$$

This problem has the same form as the combinatorial optimization problems studied in Chapter 2. However, rather than maximizing a positive definite form on $\{-1, 1\}^n$, we are minimizing a positive definite form on the non-convex set $\{-1, 1\}^n$ and we cannot produce a γ-approximation for any reasonable γ. As we have seen, such an approximation would provide guarantees on solving NPP. On the other hand, the random rounding procedure still can produce primal feasible $\mathbf{u}_M$ with good clustering performance. Furthermore, even though we cannot produce a γ-approximation, we can estimate how well such a procedure is working by comparing the primal cost of the sampled $\mathbf{u}_M$ to the dual bound provided by (4.25). If a sampled $\mathbf{u}_M$ achieves the dual optimal, then we have produced a primal optimal $\mathbf{u}_M$.

To review, the semidefinite relaxation of (4.24) is given by the semidefinite program

$$\begin{aligned} \min_{\mathbf{Z}} \quad & \mathrm{Tr}\left(\begin{bmatrix} T & \mathbf{y}_p^\top \\ \mathbf{y}_p & \mathbf{Q}_{MM} \end{bmatrix} \mathbf{Z}\right) \\ \text{s.t.} \quad & Z_{ii} = 1 \\ & \mathbf{Z} \succeq 0 \end{aligned} \tag{4.25}$$

with $\mathbf{Z}$ an $M+1 \times M+1$ matrix. The randomized approximation algorithm would proceed as follows

(i) Solve the semidefinite program (4.25) to yield an optimal positive semidefinite matrix $\mathbf{Z}$.

(ii) Sample $\mathbf{x}$ from a gaussian with mean 0 and covariance $\mathbf{Z}$.

(iii) Set $[\mathbf{z}, \mathbf{u}_M] = \mathrm{sign}(\mathbf{x})\,\mathrm{sign}(x_1)$. This guarantees $z = 1$.

(iv) Continue to sample until the cost of $\mathbf{u}$ in (4.24) is sufficiently small.

When there are no labels, and the constant functions are elements in the RKHS, the trivial labelling $f(\mathbf{x}_i) = 1$ is often the lowest cost solution to the least-squares clustering problem. To avoid this trivial solution, we may to adjoin the constraint $\sum_{i=1}^N f(\mathbf{x}_i) = 0$ to the optimization. Unfortunately, the Tikhonov cost is no longer of the form (4.23). We can solve the Tikhonov regularization problem with this new constraint as follows. Consider

$$\begin{aligned} \min_f \quad & \sum_{i=1}^N (f(\mathbf{x}_i) - y_i)^2 + \lambda \|f\|_K^2 \\ \text{s.t.} \quad & \sum_{i=1}^N f(\mathbf{x}_i) = 0 \end{aligned} \tag{4.26}$$

The dual problem is now given by

$$\min_{\mathbf{c},d} \lambda \mathbf{c}^\top \mathbf{c} - 2\mathbf{y}^\top \mathbf{c} + (\mathbf{c} + d\mathbf{1})^\top \mathbf{K} (\mathbf{c} + d\mathbf{1})^\top \tag{4.27}$$

which can be solved to give the optimal Tikhonov cost

$$\mathbf{y}^\top(\hat{\mathbf{K}} + \lambda \mathbb{1})\mathbf{y} \tag{4.28}$$

with

$$\hat{\mathbf{K}} = \mathbf{K} - \frac{\mathbf{K}\mathbf{1}\mathbf{1}^\top\mathbf{K}}{\mathbf{1}^\top\mathbf{K}\mathbf{1}} \tag{4.29}$$

That is, adding the balancing constrain amounts to shifting the kernel matrix.

This shift has an intuitive explanation in the RKHS: it amounts projecting kernels centered at the data onto the space orthogonal to their mean $\mathbf{m} = \frac{1}{N}\sum_{i=1}^{N} k(\mathbf{x}_i, \cdot)$. To see this, define the linear operator

$$\mathbf{P}\mathbf{v} = \mathbf{v} - \frac{\langle \mathbf{m}, \mathbf{v} \rangle}{\langle \mathbf{m}, \mathbf{m} \rangle}\mathbf{m}. \tag{4.30}$$

$\hat{\mathbf{K}}$ is readily seen to be the Gram matrix of the functions $\mathbf{P}k(\mathbf{x}_i, \cdot)$ for $i = 1, \ldots, N$. Note that $\mathbf{P}^2 = 1$ and compute the Gram matrix as the infinite dimensional matrix outer product

$$\hat{\mathbf{K}} = [k(\mathbf{x}_1, \cdot), \ldots, k(\mathbf{x}_N, \cdot)]^\top \mathbf{P} [k(\mathbf{x}_1, \cdot), \ldots, k(\mathbf{x}_N, \cdot)] \tag{4.31}$$

Minimizing cost (4.28) with the constraints $u_i^2 = 1$ can be performed using the random rounding procedure described above. This algorithm performs quite well as demonstrated out in the experiments.

In the next section, we will show that we can also approximate the least-squares clustering problem via the generalized eigenvalues of the kernel matrix and a family diagonal matrices. These approximations will turn out to be the family of algorithms called Spectral Clustering.

4.4 Eigenvalue Approximations and the Normalized Cuts Algorithm

It is well understood that a quadratic program with exactly one quadratic constraint admits a solution by a generalized eigenvalue method. In particular, if we begin with the transductive algorithms of this chapter and produce a convex combination of the quadratic constraints, we can produce approximations to clustering and transduction that can be solved faster than standard semidefinite programs.

Let us begin with the least-squares clustering problem

$$\begin{aligned} \min_{\mathbf{u}} \quad & \mathbf{u}^\top (\hat{\mathbf{K}} + \lambda \mathbb{1})^{-1} \mathbf{u} \\ \text{s.t.} \quad & u_i^2 = 1 \end{aligned} \tag{4.32}$$

Given any α_i, we can approximate this problem with the optimization

$$\begin{aligned} \min_{\mathbf{u}} \quad & \mathbf{u}^\top (\hat{\mathbf{K}} + \lambda \mathbb{1})^{-1} \mathbf{u} \\ \text{s.t.} \quad & \sum_{i=1}^{N} \alpha_i u_i^2 = \sum_{i=1}^{N} \alpha_i \end{aligned} \tag{4.33}$$

Or, equivalently in matrix form

$$\begin{aligned} \min_{\mathbf{u}} \quad & \mathbf{u}^\top (\hat{\mathbf{K}} + \lambda \mathbb{1})^{-1} \mathbf{u} \\ \text{s.t.} \quad & \mathbf{u}^\top \operatorname{diag}(\alpha) \mathbf{u} = \mathbf{1}^\top \alpha \end{aligned} \tag{4.34}$$

This problem can be solved by finding the largest generalized eigenvalue of the pair of matrices $\hat{\mathbf{K}} + \lambda \mathbb{1}, \operatorname{diag}(1/\alpha))$. By choosing different α, we get different algorithms.

Before showing that this can reproduce Normalized Cuts, let us first show that the bounds of the primal clustering cost produced by these eigenvalue relaxations are always more conservative than the semidefinite programming relaxations.

Theorem 4.4.1 *The dual of (4.32) achieves a higher cost than the relaxation (4.34).*

Proof The dual of (4.32) is the semidefinite program (see Chapter 2):

$$\begin{aligned} \max_{\gamma} \quad & \sum_{i=1}^{N} \gamma_i \\ \text{s.t.} \quad & (\hat{\mathbf{K}} + \lambda \mathbb{1})^{-1} - \text{diag}(\gamma) \succeq 0 \end{aligned} \tag{4.35}$$

The dual of (4.34) is:

$$\begin{aligned} \max_{\beta} \quad & \beta \\ \text{s.t.} \quad & (\hat{\mathbf{K}} + \lambda \mathbb{1})^{-1} - \beta \, \text{diag}(\alpha) \succeq 0 \end{aligned} \tag{4.36}$$

There is no duality gap between (4.34) and this dual (4.36). The dual (4.35) optimizes over an arbitrary diagonal matrix, whereas the dual (4.36) optimizes over a more constrained diagonal matrix. Therefore (4.34) is a lower bound on the dual (4.35) of the clustering SVM problem. Since both of these are lower bounds on the primal clustering SVM problem (4.32), solving (4.35) yields a value closer to the optimum of (4.32) than does solving (4.34). ■

4.4.1 The Normalized Cuts Algorithm

This section provides a brief review of the Normalized Cuts algorithm. Given a set of data points $\mathbf{x} = \{x_i | x_i \in \mathcal{R}^d, i \in 1..N\}$, and an "affinity" measure $k(x, y)$, build the affinity matrix $\mathbf{K}$ with $K_{ij} = k(x_i, x_j)$. A common choice for k is the Gaussian kernel $k(x, y) = \exp\left(-\frac{\|x_i - y_i\|^2}{2\sigma^2}\right)$. The affinity matrix $\mathbf{K}$ defines the weights on a fully connected graph where each node corresponds to a data point x_i and K_{ij} is the weight of the edge between node i and node j. Assigning each x_i a label $y_i \in \{-1, +1\}$ cuts the graph into a set A of the vertices with label -1 and a set B of vertices with labels $+1$. The cost $\text{cut}(A, B)$ is the sum of the weight of the edges between vertices in A and vertices in B. The goal of Normalized Cuts [90] is to find the cut that minimizes the following cost function:

$$\text{cut}(A, B) \left(\frac{1}{\text{Vol}(A)} + \frac{1}{\text{Vol}(B)} \right), \tag{4.37}$$

where Vol is the sum of the weights in a set. This cost function is designed to penalize cuts that are not well balanced. Finding the optimal Normalized Cut is NP hard, so the Normalized Cuts algorithm optimizes a relaxation of the above:

$$\begin{aligned} \mathbf{v}^* = \quad & \arg\max_{\mathbf{v}} \quad \frac{\mathbf{v}^\top \mathbf{D}^{-\frac{1}{2}} \mathbf{K} \mathbf{D}^{-\frac{1}{2}} \mathbf{v}}{\mathbf{v}^\top \mathbf{v}} \\ & \text{s.t.} \qquad \mathbf{v}^\top \mathbf{D}\mathbf{1} = 0 \end{aligned}$$

$\mathbf{D}$ is a diagonal matrix whose iith entry is the sum of the ith row of $\mathbf{K}$, and $\mathbf{1}$ is the column vector of all ones. The optimum $\mathbf{v}$ is the second eigenvector of $\mathbf{D}^{-\frac{1}{2}} \mathbf{K} \mathbf{D}^{-\frac{1}{2}}$ (we casually refer to the nth eigenvector of a matrix as a shorthand for the eigenvector corresponding to the nth largest eigenvalue). The components of $\mathbf{v}^*$ are then thresholded to yield a vector in $\{-1, +1\}^N$:

$$\hat{y} \quad = \quad \text{sgn}(\mathbf{v}^*). \tag{4.38}$$

This is the labeling as reported by Normalized Cuts. We refer to this algorithm as the Normalized Cuts algorithm (or just Normalized Cuts) and the unrelaxed cost function (4.37) as the Normalized Cut cost. Other relaxations for (4.37) are possible [108], but we do not provide an interpretation for these relaxations here.

Of course, since $\mathbf{K}$ is a kernel matrix, we should immediately suspect that there is a tie-in with the theory of reproducing kernel Hilbert spaces. Indeed, if we set $\alpha_i = 1/(d_i)$ in (4.34), then as λ goes to zero, we recover normalized cuts as an eigenvalue approximation to the least-squares clustering problem.

To see the equivalence, consider the generalized eigenvalue equation

$$\hat{\mathbf{K}}\mathbf{v} = \mu \, \text{diag}(1/\alpha)\mathbf{v} = \mu \mathbf{D}\mathbf{v} \tag{4.39}$$

It is easy to see that $\mathbf{1}$ is a generalized eigenvector of $\mathbf{K}$ and $\mathbf{D}$, and it is shown in [90] that it corresponds to the largest eigenvalue. Furthermore, any other generalized

eigenvector is orthogonal to $\mathbf{D1} = \theta\mathbf{K1}$. Now

$$\hat{\mathbf{K}}\mathbf{1} = \mathbf{K1} - \frac{\mathbf{K1}\mathbf{1}^\top\mathbf{K}}{\mathbf{1}^\top\mathbf{K1}}\mathbf{1} = 0 \tag{4.40}$$

and if $\mathbf{v}$ is another generalized eigenvector corresponding to eigenvalue μ, then

$$\hat{\mathbf{K}}\mathbf{v} = \mathbf{Kv} - \frac{\mathbf{K1}\mathbf{1}^\top\mathbf{K}}{\mathbf{1}^\top\mathbf{K1}}\mathbf{v} = \mathbf{Kv} - \frac{\mathbf{K1}}{\mathbf{1}^\top\mathbf{K1}}(\frac{1}{\theta}\mathbf{1}^\top\mathbf{Dv}) = \mathbf{Kv} = \mu\mathbf{Dv} \tag{4.41}$$

so the largest generalized eigenvector of $\hat{\mathbf{K}}$ and $\mathbf{D}$ is $\mathbf{v}^*$, the second eigenvector of $\hat{\mathbf{K}}$ and $\mathbf{D}$.

The particular choice of the weight factor $1/d_i$ in is an implicit design choice in the Normalized Cuts algorithm. It gives greater weight to points with low affinity to the remainder of the data set. This weighting appears to make Normalized Cuts sensitive to outliers, which is undesirable. The only benefit we see in this weighting is that finding a solution to equation (4.34) is simplified, because the second eigenvector of automatically satisfies $\mathbf{1Dv} = 0$. Other weightings are possible and will be explored in later sections.

Using the representer form, we can produce the function that Normalize cuts uses to label space. By plotting the zero-contour of such a function, we can illustrate some of the weaknesses of the algorithm. Figure 4-1 demonstrates Normalized Cuts' sensitivity to an outlier. By sliding one outlier along the x-axis, the clustering boundary can be arbitrarily shifted to the left or to the right. Figure 4-2 shows that Normalized Cuts will split elongated structures, because according to its weighting, it is favorable to have points on opposite ends of an elongated structure land on opposite sides of the separating plane.

4.4.2 Average Gap Algorithm

If equal weight is given to every point, the outlier and splitting problems are attenuated. Set $\alpha_i = 1/N$ in Equation (4.34). This shall be referred to as the *Average Gap Algorithm* [80]. In this case,the quadratic constraint becomes $\sum_{i=1}^{N} u_i = N$. In this

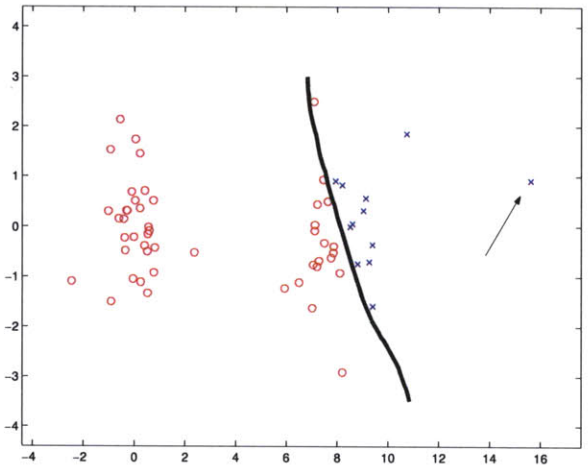

Figure 4-1: In Normalized Cuts, an outlier can dwarf the influence of other points, because points away from the mean are heavily weighted. Sliding the outlier (indicated by the arrow) along the x-axis can shift the clustering boundary arbitrarily to the left or the right. Without the outlier, Normalized Cuts places the boundary between the two clusters.

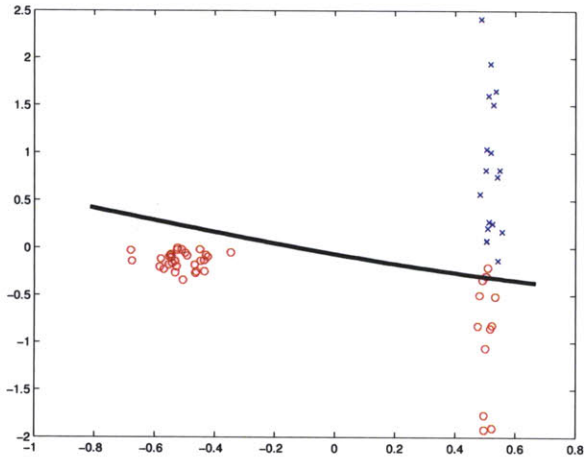

Figure 4-2: Because Normalized Cuts puts more weight on points away from the mean, it prefers to have the ends of the elongated vertical cluster on opposite sides of the separating hyperplane.

case, we only need to compute the maximum eigenvalue of $\hat{\mathbf{K}}$. This is quite similar to the Factorization Approach proposed by Perona and Freeman [76], but, crucially, using the eigenvectors $\hat{\mathbf{K}}$ instead of $\mathbf{K}$

Proposition 4.4.2 *The label assignments from the average gap algorithm are* $\mathrm{sgn}(v)$ *where* v *is the largest eigenvector of* $\hat{\mathbf{K}}$.

Proof By setting $\alpha_i = 1/N$, we are left with computing the largest generalized eigenvector, $\mathbf{v}$, of $\hat{\mathbf{K}}$ and $\mathbb{1}$. This is, of course, just the largest eigenvector of $\hat{\mathbf{K}}$. The labelling function is given by

$$f(\mathbf{x}) = \sum_{i=1}^{N} c_i k(\mathbf{x}_i, \mathbf{x}) \tag{4.42}$$

with $\mathbf{Kc} = \mathbf{v}$. Then the labels of the data are given by

$$\hat{y}_i = \mathrm{sgn}(f(\mathbf{x}_i)) = \mathrm{sgn}([\mathbf{Kc}]_i) = \mathrm{sgn}(v_i) \tag{4.43}$$

■

Section 4.5 shows that this new hyperplane algorithm works as well as Normalized Cuts, and is less susceptible to outliers. We will refer to it as the Average Gap algorithm. Compare Figure 4-3 with Figure 4-1. The outlier does not affect the clustering boundary, no matter how far it is from the main body. Adding several outliers eventually does move the boundary (not shown). Also compare Figure 4-4 with Figure 4-2. The elongated cluster is not split up. No stretching of the elongated cluster causes it to be split.

4.5 Numerical Experiments

We compared Normalized Cuts (NCUT), the Average Gap (AVGGAP) algorithm, and the semidefinite relaxation of least-squares clustering (CSDP) on datasets from the UCI repository [20]. The Wisconsin breast cancer data set (`cancer1`) and the new diagnostic dataset (`cancer2`) were originally obtained from the University of

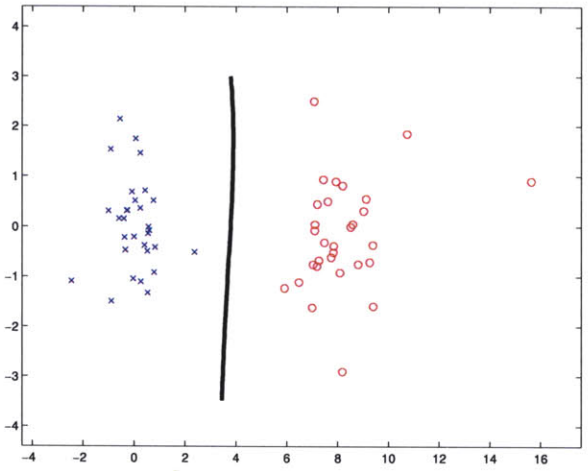

Figure 4-3: The data set of Figure 4-1 is correctly segmented by weighting all points equally. The outlier point doesn't shift the clustering boundary significantly.

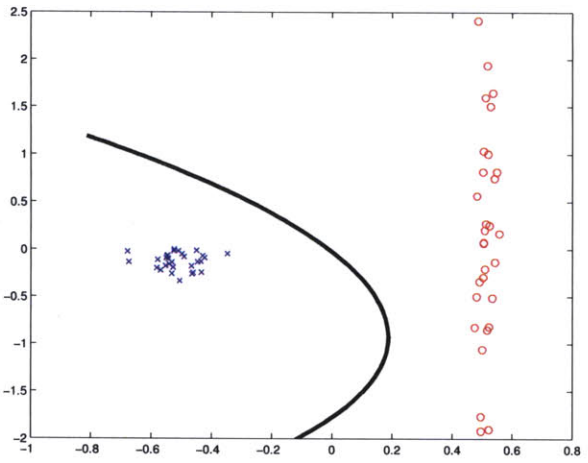

Figure 4-4: The data set of Figure 4-2 is correctly segmented by weighting all points equally.

dataset	n	CSDP linear	CSDP gaussian	AVGGAP linear	AVGGAP gaussian	NCUT
`wine`	130	100	100	56	86	97
`cancer1`	683	91	91	91	91	91
`cancer2`	569	97	98	97	97	97
`ionosphere`	351	70	90	60	90	57

Table 4.1: Clustering performance.

Wisconsin Hospitals, Madison. We also used the first two classes in the wine recognition dataset (`wine`) and the ionosphere dataset (`ionosphere`). In all of data sets, clustering performance is the percentage of correctly assigned labels to each class. Because of the balancing constraint $\sum_{i=1}^{N} f(\mathbf{x}_i) = 0$, none of the algorithms assigned the same label to the entire data set.

The semidefinite program for the least-squares clustering was solved using the bundle method [40]. We were able to run sets of one hundred points in a few seconds and sets of two thousand points in about five minutes on a dual Xeon 2.8 GHZ machine.

The clustering algorithms were all compared using a gaussian with kernel parameter tuned such that $C = 2 * max_{i,j}(\|x_i - x_j\|)$. The CSDP and AVGGAP algorithms were also run using linear kernels. The Normalized cuts algorithm cannot cluster using a linear kernel as it requires the kernel to be everywhere non-negative. Table 4.5 reports the performance of the algorithms. In all cases, the SDP does as well or better than the other algorithms. In some instances, like in the wine or ionosphere data sets, the eigenvalue relaxations may perform quite poorly.

4.6 Conclusion

Due to the binary constraints, transduction, clustering, and segmentation are all hard combinatorial problems. We presented two families of approximations using Lagrangian duality.

We have provided a function learning interpretation for the Normalized Cuts re-

laxation of Shi and Malik and showed that it can be thought of as an approximation to regularized least-squares clustering. Our interpretation of Normalized Cuts can also be used to justify semi-supervised versions of it as well, although we did not explore this possibility in this chapter. In fitting this function, Normalized Cuts pays more attention to outliers, and so fails to recover sensible clusters in some cases. We showed how to avoid this pitfall by weighting all data points equally. The regularized least-squares clustering problem is better approximated by the semidefinite approximation.

Chapter 5

Output Prior: Dynamics

Learning the mapping between samples of two different time series is a ubiquitous application of function fitting. For example, in tracking, one transforms a time series of observations from sensors to the pose of a target; one can generate computer animation by transforming a time series representing the motions of an actor to vectorized graphics; and system-identification learns the transformation between the inputs and outputs of a plant. Depending on the details of the application, different domain-specific information is leveraged to design particular algorithms, but exploiting latent dynamics models improves the performance of all such algorithms.

In this chapter, we consider consequences of imposing dynamics priors on the labels $\mathbf{u}$. Such priors enable the transformation a variety of time series with surprisingly few output examples. The main contribution is a synthesis of a semi-supervised regression model that takes finds a function whose outputs are consistent with with physical dynamics defined by a linear-Gaussian Markov chain. The optimization allows a user to label a few data points to specify a coordinate system and to provide guidance to the algorithm when needed.

We demonstrate the utility of these optimization methods with an interactive tracking system where the user specifies a desired output for a few key samples in a time series of sensor measurements. These examples, together with the unlabelled portion of the time series, allow the system to compute a function that maps as-yet unseen measurements to the desired representation. Using synthetic data, we will

show that this interactive tracker can be used to perform transformations of manifold learning when the data is time ordered. We also demonstrate the tracker on different real-world examples as well. An articulated body tracking experiment where the user specifies positions of the subject's limbs in a video sequence illustrates the robustness of the framework even when no video pre-processing is applied. We show how to calibrate a HCI device by transforming the voltages induced in a set of antennae by a Radio Frequency ID (RFID) tag to the position of the tag with only four labelled examples and with no labels at all.

The algorithms operate on the sensor data directly and do not require any pre-processing. In particular, we are able to recover excellent representation with very sparse sampling: in the video example, the input examples are raw pixel values of 640x480 images and we can learn a tracker with 12 examples. Furthermore, in the sensor network experiments, we show that our unsupervised algorithm can recover an excellent approximation of the target trajectory, up to scaling and orientation, with no examples whatsoever.

5.1 Related Work

Manifold learning techniques [97, 85, 8, 27, 106, 16] find a low-dimensional representation that preserves some local geometric attribute of the high-dimensional observations. This requires identifying data points that lie in a local neighborhood along the manifold around every high-dimensional data point. When the manifold is sparsely sampled, these neighboring points are difficult to identify, and the algorithms can fail to recover any meaningful structure. Our algorithm obviates the need to search for such neighbors by utilizing the time ordering of data points instead. Jenkins and Mataric [48] suggest artificially reducing the distance between temporally adjacent points to provide an additional hint to Isomap about the local neighborhoods of image windows. We also take advantage of dynamics in the low-dimensional space to allow our algorithm to better estimate the distance between pairs of temporally adjacent points along the manifold. This requires only fine enough sampling over time

to retain the temporal coherence between video frames, which is much less onerous than the sampling rate required to correctly estimate neighborhood relationships in traditional manifold learning algorithms. While various semi-supervised extensions to manifold learning algorithms have been proposed [42, 78], these algorithms still do not take advantage of the temporal coherence between adjacent samples of the input time series.

The semi-supervised regression approaches of [109] and [9] take into account the manifold structure of the data. But they also rely on brittle estimates of the neighborhood structure, and do not take advantage of the time ordering of the data set. These semi-supervised regression methods are similar to our method in that they also impose a random field on the low-dimensional representation. The work presented here augments these techniques by introducing the temporal dependency between output samples in the random field. It can be viewed as a special case of estimating the parameters of a continuously-valued conditional random field [55] or a manifold learning algorithm based on function estimation [93].

Nonlinear system identification (see [34, 98] and references within) provides another framework for introducing dynamics into manifold learning. In this context, the frames in the video are modeled as observations generated by a Markov chain of low-dimensional states. Nonlinear system identification recovers the parameters of this model, including an observation function which maps low-dimensional states to images. This usually requires approximate coordinate ascent over a non-convex space, making the algorithms computationally intensive and susceptible to local minima. Dynamic Textures [28] sidesteps these issues by performing linear system identification instead, which limits it to linear appearance manifolds. Instead of searching for a mapping from states to images, as would be done in nonlinear system identification, we search for a mapping from images to states. This results in an optimization problem that is quadratic in the latent states and the parameters of the projection function, making the problem computationally tractable and not subject to local minima.

5.2 Model for Semi-Supervised Nonlinear System ID

Let us assume that we are presented with a time series $\mathbf{x}_t \in \mathbb{R}^D$ and we want to learn a memoryless mapping $f : \mathbb{R}^D \to \mathbb{R}^d$ such that $f(\mathbf{x}_t) = \mathbf{u}_t$ and the output time series $\mathbf{u}_t$ satisfies a prescribed dynamics model. We will consider two scenarios. In the semi-supervised setting, we suppose that for a few samples $t \in S$, we are given desired values for the $\mathbf{u}_t = \mathbf{y}_t$ outputs. In the unsupervised setting, we will only assume that the $\mathbf{u}_t$ values agree with the expected values implied by the dynamics model.

Consider each component f^i of $f = [f^1(\mathbf{x}) \dots f^d(\mathbf{x})]$ separately. If the desired output of f^i were known to be $\mathbf{y}_t^i$, we could use the standard Tikhonov regularization on a RKHS to solve for the best approximation of f^i:

$$\min_{f^i} \sum_{t=1}^{T} \|f^i(\mathbf{x}_t) - \mathbf{y}_t^i\|^2 + \lambda_k \|f^i\|_k^2. \tag{5.1}$$

But in our scenario, none or only a few $\mathbf{y}_t^i$ labels are provided by the user.

Let us assume that we know *a priori* that the time series of the u_i^t is the output of a linear Gaussian time invariant system

$$\begin{aligned} \mathbf{z}_{t+1} &= \mathbf{A}\mathbf{z}_t + \omega_t \\ \mathbf{u}_t &= \mathbf{C}\mathbf{z}_t + \nu_t \end{aligned} \tag{5.2}$$

The Gaussian random variables ω_t and ν_t have zero-mean and known covariance matrices Λ_ω and Λ_ν respectively. The matrices $\mathbf{A}$, $\mathbf{C}$ and Λ_ω, and Λ_ν specify the desired dynamics, and are parameters of the optimization. Furthermore, we assume that the sample $\mathbf{x}_0$ agrees with the stationary dynamics of the linear system. Certainly, the mean of $\mathbf{x}_0$ is 0. To compute its covariance, Λ_0, we assume that it is observed in the

stationary regime such that

$$\begin{aligned}\Lambda_0 = \mathbb{E}[\mathbf{z}_t\mathbf{z}_t^\top] &= \mathbb{E}[\mathbf{z}_{t+1}\mathbf{z}_{t+1}^\top] \\ &= \mathbb{E}[(\mathbf{A}\mathbf{z}_t + \omega_t)(\mathbf{A}\mathbf{z}_t + \omega_t)^\top] \\ &= \mathbb{E}[\mathbf{A}\mathbf{z}_t\mathbf{z}_t^\top\mathbf{A}^\top + \mathbf{A}\mathbf{z}_t\omega_t^\top + \omega_t\mathbf{z}_t^\top\mathbf{A}^\top + \omega_t\omega_t^\top] \\ &= \mathbb{E}[\mathbf{A}\mathbf{z}_t\mathbf{z}_t^\top\mathbf{A}^\top + \omega_t\omega_t^\top] \\ &= \mathbf{A}\mathbb{E}[\mathbf{z}_t\mathbf{z}_t^\top]\mathbf{A}^\top + \Lambda_\omega \\ &= \mathbf{A}\Lambda_0\mathbf{A}^\top + \Lambda_\omega\end{aligned} \tag{5.3}$$

That is, Λ_0 is the solution to the discrete Lyapunov equation

$$\mathbf{A}\Lambda_0\mathbf{A}^\top - \Lambda_0 + \Lambda_\omega = 0 \tag{5.4}$$

We can compensate for the absence of labels at every data point by forcing $f^i(\mathbf{x}_t)$ to agree with the position component of the corresponding $\mathbf{u}_t$ using additional penalty terms. The semi-supervised optimization is given by

$$\begin{aligned}\min_{f,\mathbf{u},\mathbf{z}} \quad & \sum_{i=1}^{d}\sum_{t=1}^{T} \|f^i(\mathbf{x}_t) - u_t^i\|^2 + \lambda_k\|f^i\|_k^2 \\ & + \lambda_d\left(\sum_{t=2}^{T}\|\mathbf{z}_t - \mathbf{A}\mathbf{z}_{t-1}\|_{\Lambda_\omega}^2 + \mathbf{z}_1^\top\Lambda_0^{-1}\mathbf{z}_1 + \sum_{t=1}^{T}\|\mathbf{u}_t - \mathbf{C}\mathbf{z}_t\|_{\Lambda_\nu}^2\right) \\ \text{s.t.} \quad & \mathbf{u}_t = \mathbf{y}_t \quad \text{for } t \in S\end{aligned} \tag{5.5}$$

The first line is just a Tikhonov Regularization penalty on all of the component functions of f. The second line favors functions whose outputs could have been produced by the LTI system (5.27). Indeed, when all of the $\mathbf{u}_t$ are given, iteratively minimizing the second cost function over $\mathbf{z}_t$ recovers the well-known Kalman Filter. The regularization parameter λ_d may be tuned to reflect how much one trusts the dynamics model. The equality constraint enforces the given labels for the given samples in S.

In the unsupervised case, we can again appeal to Kernel PCA as described in

Chapter 3. In the limit of being driven by whitenoise, the u_t emitted from the system (5.27) will be colored white noise with statistics:

$$\begin{aligned} \mathbb{E}[\mathbf{u}_t] &= 0 \\ \mathbb{E}[\mathbf{u}_t \mathbf{u}_t^\top] &= \Lambda_{\mathbf{u}} \end{aligned} \tag{5.6}$$

Λ_y can be directly computed from Λ_0

$$\begin{aligned} \mathbb{E}[\mathbf{y}_t \mathbf{y}_t^\top] &= \mathbb{E}[(\mathbf{C}\mathbf{z}_t + \nu_t)(\mathbf{C}\mathbf{z}_t + \nu_t)^\top] \\ &= \mathbb{E}[(\mathbf{C}\mathbf{z}_t \mathbf{z}_t^\top \mathbf{C}^\top + \nu_t \nu_t^\top] \\ &= \mathbf{C}\Lambda_0 \mathbf{C}^\top + \Lambda_\nu \end{aligned} \tag{5.7}$$

And our unsupervised cost is

$$\begin{aligned} \min_{f,\mathbf{u},\mathbf{z}} \quad & \sum_{i=1}^{d} \sum_{t=1}^{T} \|f^i(\mathbf{x}_t) - u_t^i\|^2 + \lambda_k \|f^i\|_k^2 \\ & + \lambda_d \left(\sum_{t=2}^{T} \|\mathbf{z}_t - \mathbf{A}\mathbf{z}_{t-1}\|_{\Lambda_\omega}^2 + \mathbf{z}_1^\top \Lambda_0^{-1} \mathbf{z}_1 + \sum_{t=1}^{T} \|\mathbf{u}_t - \mathbf{C}\mathbf{x}_t\|_{\Lambda_\nu}^2 \right) \\ \text{s.t.} \quad & \sum_{t=1}^{T} \mathbf{u}_t = 0 \\ & \frac{1}{T} \sum_{t=1}^{T} \mathbf{u}_t \mathbf{u}_t^\top = \Lambda_{\mathbf{u}} \end{aligned} \tag{5.8}$$

These cost functions fit right into the prior on output framework. In particular, the optimal f will be given by weighted sum of kernels centered at each $\mathbf{x}_t$:

$$f^i(\mathbf{x}) = \sum_{t=1}^{T} c_t^i k(\mathbf{x}, \mathbf{x}_t)\,, \tag{5.9}$$

where the vector c^i contains the coefficients for the ith dimension of f.

The cost function of optimizations (5.5) and (5.8) is quadratic in coefficiencts of the kernel expansion for f, $\mathbf{c}$ and in the hidden states of the LTI system, $\mathbf{z}$. Since

these quantities are unconstrained, we can minimize the cost function directly. Let

$$\begin{aligned}\mathbf{S}(\mathbf{u}) := \min_{f,\mathbf{z}} \quad & \sum_{i=1}^{d}\sum_{t=1}^{T} \|f^i(\mathbf{x}_t) - y_t^i\|^2 + \lambda_k \|f^i\|_k^2 \\ & + \lambda_d \left(\sum_{t=2}^{T} \|\mathbf{z}_t - \mathbf{A}\mathbf{z}_{t-1}\|_{\Lambda_\omega}^2 + \mathbf{z}_1^\top \Lambda_0^{-1} \mathbf{z}_1 + \sum_{t=1}^{T} \|\mathbf{u}_t - \mathbf{C}\mathbf{x}_t\|_{\Lambda_\nu}^2 \right)\end{aligned} \tag{5.10}$$

We will now show that $\mathbf{S}(\mathbf{u})$ is a positive definite quadratic form in y. Indeed, we can split this cost in half:

$$\begin{aligned}\mathbf{S}(\mathbf{u}) &= \min_{f} \sum_{i=1}^{d}\sum_{t=1}^{T} \|f^i(\mathbf{x}_t) - u_t^i\|^2 + \lambda_k \|f^i\|_k^2 \\ &\quad + \lambda_d \min_{\mathbf{z}} \left(\sum_{t=2}^{T} \|\mathbf{z}_t - \mathbf{A}\mathbf{z}_{t-1}\|_{\Lambda_\omega}^2 + \mathbf{z}_1^\top \Lambda_0^{-1} \mathbf{z}_1 + \sum_{t=1}^{T} \|\mathbf{u}_t - \mathbf{C}\mathbf{x}_t\|_{\Lambda_\nu}^2 \right) \\ &=: \mathcal{T}(\mathbf{u}) + \mathcal{K}(\mathbf{u})\end{aligned} \tag{5.11}$$

We have shown that

$$\mathcal{T}(\mathbf{y}) = \lambda_r \sum_{i=1}^{d} \mathbf{y}^{i\top} (\mathbf{K} + \lambda_r \mathbb{1})^{-1} \mathbf{y}^i \tag{5.12}$$

The second term can also be written down explicitly. First, define the $NT \times NT$ matrices

$$\begin{aligned}\hat{\mathbf{A}} &:= \begin{bmatrix} \mathbf{A} & -\mathbb{1} & 0 & \cdots & \\ 0 & \mathbf{A} & -\mathbb{1} & 0 & \cdots \\ & & \ddots & & \\ 0 & \cdots & 0 & \mathbf{A} & -\mathbb{1} \\ 0 & \cdots & 0 & 0 & \mathbb{1} \end{bmatrix} \\ \hat{\Lambda} &:= \operatorname{diag}(\mathbb{1}_{T-1} \otimes \Lambda_\omega, \Lambda_0)\end{aligned} \tag{5.13}$$

and note

$$\begin{aligned}\mathcal{K}(\mathbf{u}) &= \min_{\mathbf{z}} \begin{bmatrix} \mathbf{u} \\ \mathbf{z} \end{bmatrix}^\top \begin{bmatrix} \mathbb{1} \otimes \Lambda_\nu^{-1} & \mathbb{1}_T \otimes (\Lambda_\nu^{-1}\mathbf{C}) \\ \mathbb{1}_T \otimes (\mathbf{C}^\top \Lambda_\nu^{-1}) & \hat{\mathbf{A}}\hat{\Lambda}^{-1}\hat{\mathbf{A}} + \mathbb{1}_T \otimes (\mathbf{C}^\top \Lambda_\nu^{-1}\mathbf{C}) \end{bmatrix} \begin{bmatrix} \mathbf{u} \\ \mathbf{z} \end{bmatrix} \\ &= \mathbf{u}^\top \left(\mathbb{1} \otimes \Lambda_\nu^{-1} - \mathbb{1}_T \otimes (\Lambda_\nu^{-1}\mathbf{C})(\hat{\mathbf{A}}\hat{\Lambda}^{-1}\hat{\mathbf{A}} + \mathbb{1}_T \otimes (\mathbf{C}^\top \Lambda_\nu^{-1}\mathbf{C}))^{-1} \mathbb{1}_T \otimes (\mathbf{C}^\top \Lambda_\nu^{-1}) \right) \mathbf{u} \\ &= \mathbf{u}^\top \left(\mathbb{1}_T \otimes \Lambda + (\mathbb{1}_T \otimes \mathbf{C})(\hat{\mathbf{A}}\hat{\Lambda}^{-1}\hat{\mathbf{A}})^{-1}(\mathbb{1}_T \otimes \mathbf{C}^\top) \right)^{-1} \mathbf{u}. \end{aligned} \tag{5.14}$$

The first equality is the standard solution to the quadratic program, and the second equality follows from the matrix inversion lemma. Both results are derived in Appendix A.

While this cost function looks a bit unwieldy, we can simplify it by realizing that the quadratic form is nothing more than the inverse of the joint covariance of the $\mathbf{u}_t$

$$\Lambda^{\mathbf{u}} := \left(\mathbb{1}_T \otimes \Lambda_\nu + (\mathbb{1}_T \otimes \mathbf{C})(\hat{\mathbf{A}}\hat{\Lambda}^{-1}\hat{\mathbf{A}})^{-1}(\mathbb{1}_T \otimes \mathbf{C}^\top) \right) \tag{5.15}$$

In turn, we can use the Gaussian statistics of the LTI system to compute $\Lambda^{\mathbf{u}}$

Proposition 5.2.1 *$\Lambda^{\mathbf{u}}$ is a symmetric, positive-definite, block Toeplitz matrix and for $s \geq t$*

$$\Lambda^{\mathbf{u}}_{st} = \mathbf{C}\mathbf{A}^{s-t}\Lambda_0\mathbf{C}^\top + \delta_{st}\Lambda_\nu \tag{5.16}$$

Proof Since

$$\mathbb{E}[\mathbf{u}_s\mathbf{u}_t^\top] = \mathbf{C}\mathbb{E}[\mathbf{x}_s\mathbf{x}_t^\top]\mathbf{C}^\top + \delta_{st}\Lambda_\nu \tag{5.17}$$

we need only compute $\mathbb{E}[\mathbf{x}_s\mathbf{x}_t^\top]$. Suppose without loss of generality that $s \geq t$

$$\mathbf{x}_t = \sum_{j=0}^{t} \mathbf{A}^{t-j}\omega_j\mathbf{x}_s = \sum_{k=t+1}^{s} \mathbf{A}^{s-k}\omega_k + \mathbf{A}^{s-t}\mathbf{x}_t \tag{5.18}$$

Since ω_k is i.i.d, $\mathbb{E}[\mathbf{x}_s\mathbf{x}_t] = \mathbb{E}[\mathbf{A}^{s-t}\mathbf{x}_t\mathbf{x}_t] = \mathbf{A}^{s-t}\mathbb{E}[\mathbf{x}_t\mathbf{x}_t] = \mathbf{A}^{s-t}\Lambda_0$. This proves the proposition. ■

We may now combine the two quadratic forms to yield a joint optimization over $\mathbf{u}$

$$\mathcal{S}(\mathbf{u}) = \lambda_r \mathbf{u}^\top \left(\mathbf{K} \otimes \mathbb{1}_d + \lambda_r \mathbb{1}_{dT}\right))^{-1} \mathbf{u} + \lambda_d \mathbf{u}^\top \Lambda^{\mathbf{u}-1} \mathbf{u} =: \mathbf{u}^\top \mathbf{S} \mathbf{u} \tag{5.19}$$

where $\mathbf{S}$ is the sum of two positive definite quadratic forms and is hence positive definite. In this regard, it now becomes apparent how to solve for the $\mathbf{u}$ labels.

In the semi-supervised case, we will be minimizing a positive definite quadratic form over a subset of the $\mathbf{u}_t$ with those $\mathbf{u}_t$ with $t \in S$ held fixed. This amounts to solving a system of linear equations. We will present a fast method to solve this particular system in the next section.

In the unsupervised case, we minimize a positive definite quadratic form subject to a linear constraint forcing the $\mathbf{u}_t$ to be zero mean and a set of quadratic constraints enforcing the second moments of the individual $\mathbf{u}_t$ variables. We will see that this problem can be solved as an eigenvalue problem when we assume that the hidden dynamical system is composed of d independent Markov chains.

Note that in both cases, we first recover the labels $\mathbf{u}$. The function $\mathbf{f}$ mapping $\mathbf{x}$'s to $\mathbf{u}$'s can be found by setting $\mathbf{c}^i = (\mathbf{K} + \lambda \mathbb{1}_T)\mathbf{u}^i$.

5.2.1 Semi-supervised Algorithm

Under the constraints that $\mathbf{u}_t = \mathbf{y}_t$ for $t \in S$, we may partition the vector $\mathbf{u}$ into the labelled components and the unlabelled components: $\mathbf{u} =: [\mathbf{y}_L \mathbf{u}_U]$. That is, $\mathbf{y}_L$ denotes the vector composed of all the $\mathbf{y}_t$ with $t \in S$ stacked on top of each other and $\mathbf{u}_U$ denotes the vector composed of all the unlabelled points with $t \notin S$. We may minimize the quadratic cost with respect to $\mathbf{u}_U$ and holding $\mathbf{y}_L$ fixed giving the well-known least squares solution

$$\mathbf{u}_U^* = -\mathbf{S}_{UU}^{-1} \mathbf{S}_{UL} \mathbf{y}_L \tag{5.20}$$

Indeed, a direct algorithm for solving the semi-supervised problem would be to compute the matrix $\mathbf{S}$, and then compute this quantity. For large T and d, this

requires four matrix inverses each of size Td which may be undesirable. Here we present an algorithm that is much faster in practice.

First note that we can write

$$\min_{\mathbf{u}_1,\mathbf{u}_2} \begin{bmatrix} \mathbf{u}_1 \\ \mathbf{y}_L \\ \mathbf{u}_2 \\ \mathbf{y}_L \end{bmatrix}^\top \begin{bmatrix} \lambda_r \left(\mathbf{K} \otimes \mathbb{1}_d + \lambda_r \mathbb{1}_{dT}\right)^{-1} & 0 \\ 0 & \lambda_d \Lambda^{\mathbf{u}-1} \end{bmatrix} \begin{bmatrix} \mathbf{u}_1 \\ \mathbf{y}_L \\ \mathbf{u}_2 \\ \mathbf{y}_L \end{bmatrix} \tag{5.21}$$

$$\text{s.t.} \quad \mathbf{u}_1 = \mathbf{u}_2$$

Introducing a Lagrange multiplier β to enforce the equality constraint, gives the Lagrangian

$$\mathcal{L} = \begin{bmatrix} \mathbf{u}_1 \\ \mathbf{y}_L \\ \mathbf{u}_2 \\ \mathbf{y}_L \end{bmatrix}^\top \begin{bmatrix} \lambda_r \left(\mathbf{K} \otimes \mathbb{1}_d + \lambda_r \mathbb{1}_{dT}\right)^{-1} & 0 \\ 0 & \lambda_d \Lambda^{\mathbf{u}-1} \end{bmatrix} \begin{bmatrix} \mathbf{u}_1 \\ \mathbf{y}_L \\ \mathbf{u}_2 \\ \mathbf{y}_L \end{bmatrix} + \beta(\mathbf{u}_1 - \mathbf{u}_2) \tag{5.22}$$

Minimizing with respect to $\mathbf{u}_1$ and $\mathbf{u}_2$ gives

$$\begin{aligned} \mathbf{u}_1 &= \hat{\mathbf{u}}_1 + \frac{1}{\lambda_r}[(\mathbf{K} \otimes \mathbb{1}_d + \lambda_r \mathbb{1}_{dT})^{-1}]^{-1}_{UU}\beta \\ \mathbf{u}_2 &= \hat{\mathbf{u}}_1 - \frac{1}{\lambda_d}[\Lambda_y^{-1}]^{-1}_{UU}\beta \\ \hat{\mathbf{u}}_1 &= [(\mathbf{K} \otimes \mathbb{1}_d + \lambda_r \mathbb{1}_{dT})^{-1}]^{-1}_{UU}[(\mathbf{K} \otimes \mathbb{1}_d + \lambda_r \mathbb{1}_{dT})^{-1}]_{UL}\mathbf{y}_L \\ \hat{\mathbf{u}}_2 &= [\Lambda^{\mathbf{y}^{-1}}]^{-1}_{UU}[\Lambda^{\mathbf{u}-1}]_{UL}\mathbf{y}_L \end{aligned} \tag{5.23}$$

That is, $\hat{\mathbf{u}}_1$ is the solution to the Tikhonov regularization problem adjusted by the Lagrange multiplier term. $\hat{\mathbf{u}}_2$ is the estimation of the outputs of the LTI system also adjusted by a Lagrange multiplier term. We can simplify these equations by employing the formula for the inverses of partitioned matrices found in Appendix A

and rewriting the last two lines as

$$\begin{aligned}\hat{\mathbf{u}}_1 &= [\mathbf{K} \otimes \mathbb{1}_d + \lambda_r \mathbb{1}_{dT}]_{UL} [\mathbf{K} \otimes \mathbb{1}_d + \lambda_r \mathbb{1}_{dT}]_{LL}^{-1} \mathbf{y}_L \\ \hat{\mathbf{u}}_2 &= \Lambda_{UL}^{\mathbf{y}} {\Lambda_{LL}^{\mathbf{y}}}^{-1} \mathbf{y}_L\end{aligned} \tag{5.24}$$

We can solve for β by setting $\mathbf{u}_1 = \mathbf{u}_2$

$$\begin{aligned}\beta &= (\frac{1}{\lambda_R}[(\mathbf{K} \otimes \mathbb{1}_d + \lambda_r \mathbb{1}_{dT})^{-1}]_{UU}^{-1} + \frac{1}{\lambda_D}[\Lambda_y^{-1}]_{UU}^{-1})^{-1}(\hat{\mathbf{u}}_2 + \hat{\mathbf{u}}_1) \\ &= (\frac{1}{\lambda_R}(\mathbf{K} \otimes \mathbb{1}_d + \lambda_r \mathbb{1}_{dT}|U) + \frac{1}{\lambda_D}(\Lambda_y|U))^{-1}(\hat{\mathbf{u}}_2 + \hat{\mathbf{u}}_1)\end{aligned} \tag{5.25}$$

where $(\mathbf{A}|U)$ denotes the Schur complement $\mathbf{A}_{UU} - \mathbf{A}_{UL}\mathbf{A}_{LL}^{-1}\mathbf{A}_{LU}$.

Thus, the algorithm works by first solving for $\hat{\mathbf{y}}_i$, then computing the Lagrange multiplier, and then adjusting the initial estimates. The first step is a least squares problem with a matrix inversion of size $|S|$ and a matrix multiplication of size $|S| - T \times T$. The second step requires computing Schur complements and solving a $|S| - T \times |S| - T$ system of linear equations. Finally, the correction step involves a system of equations with one of the Schur complements, and is again solving a $|S| - T \times |S| - T$ system of linear equations. Overall, this procedure is more efficient than directly computing the quadratic form $\mathbf{S}$ that requires computing explicit matrix inverses.

5.2.2 Unsupervised Algorithm

In the case of the unsupervised algorithm, we will now present an eigenvalue problem for when the underlying model consists of d independent Markov chains with one-dimensional outputs. In this case, we can compute cost function for each chain independently, and sum them together to yield the joint optimization

$$\begin{aligned}\min_{\mathbf{u}} \quad & \sum_{i=1}^{d} \mathbf{u}^{i\top} \left(\lambda_r(\mathbf{K} + \lambda_r \mathbb{1})^{-1} + \lambda_d \mathbf{T}^{-1}\right) \mathbf{u}^i \\ \text{s.t.} \quad & \sum_t \mathbf{u}_t = 0 \\ & \sum_t \mathbf{u}_t \mathbf{u}_t^\top = \alpha \mathbb{1}\end{aligned} \tag{5.26}$$

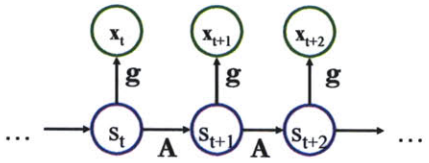

Figure 5-1: A generative model for a linear system with nonlinear output. The states $\mathbf{s}_t$ are low-dimensional representations lifted to high dimensional observables $\mathbf{x}_t$ by an embedding g.

This can be solved with an iterative eigenvalue solver as follows. First compute the matrix $\mathbf{A} = (\lambda_r(\mathbf{K} + \lambda_r \mathbb{1})^{-1} + \lambda_d \mathbf{T}^{-1})^{-1}$. Then, ensure that $\mathbf{u}^i$ has zero mean, compute the projection onto the zero-mean subspace $\mathbf{P} := \mathbb{1} - \mathbf{1}\mathbf{1}^\top / T$. Then the d largest eigenvalues of $\mathbf{PAP}$ are the optimal assignments for $\mathbf{u}^i$.

5.3 Relation to System Identification

Figure 5-1 depicts a standard generative model for time series generated by observing a Markov chain with a nonlinear objective function. The latent state evolves according to some Markov chain of states $\mathbf{z}_t$, $t = 1 \ldots T$. At each time step, a nonlinear function $g : \mathbb{R}^d \to \mathbb{R}^D$ maps a linear function of the state $\mathbf{y}_t = \mathbf{C}\mathbf{z}_t$ to a D dimensional output vector $\mathbf{x}_t$. Effects not accounted for by g are modeled as iid noise modifying the output of g.

Learning the parameters of this generative model from a sequence of observations $\mathbf{y}_1, \ldots, \mathbf{y}_T$ can be computationally expensive [34, 98] even when the dynamics of the hidden Markov chain are known. Instead of solving for g in this generative model, we recover a projection function $f : \mathbb{R}^D \to \mathbb{R}^d$ that maps images to their low-dimensional representation in a random field. This random field consists of a function f that maps the sequence of observed images to a sequence in $\mathbb{R}^d$ that evolves in accordance with a Markov chain. The random field mirrors the generative model of Figure 5-1 by modeling the interactions between a Markov chain, the observations, and supervised points provided by the user. Figure 5-2 depicts a random field describing

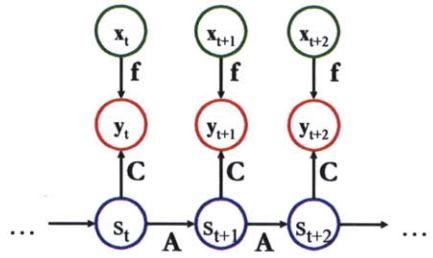

Figure 5-2: Forcing agreement between projections of observed $\mathbf{x}_t$ and a Markov chain of states $\mathbf{s}_t$. The function f maps observations to outputs of a linear system.

the factorization prescribed by (5.5).

In the case when the noise which corrupts the nonlinear output function is small, and when g is $1-1$, our algorithms approximate this standard system identification problem by searching for a pseudoinverse for g. Searching for this pseudoinverse is advantageous because, when $D >> d$, the number of functions one fits might be more than the number of samples. Furthermore, our resulting optimizations are convex, do not have local minima, and admit efficient algorithms, all of which plague the the related work that searches for g.

5.4 Interactive Tracking Experiments

To compare with Isomap, LLE and Laplacian Eigenmaps, we relied on source code available from the respective authors' web sites. We also compare against Belkin and Nyogi's graph Laplacian-based semi-supervised regression algorithm [9], which we refer to as BNR in this section. We used our own implementation of BNR.

The experiments elucidate the following features of the regression with dynamics prior. Explicitly taking into account the dynamics of the low-dimensional process obviates the need to build the brittle neighborhood graphs that are common in manifold learning and semi-supervised learning algorithms. This renders our algorithm less sensitive to errors in estimates of neighborhoods.

The assumed dynamics model does not need to be very accurate. Indeed, in what follows we will use a very simple model described below that does not match the true dynamics for any of the experiments. This model captures the time coherence

of related samples and that correlations vanish over time. Furthermore, we produce very good results with very few labelled examples that do not need to capture all the modes of variation of the data.

5.4.1 The Dynamics Model

In all of these experiments, we use an intuitive dynamics model that yields surprisingly good results. All of our applications are assumed to be tracking objects with physical mass. Because we know the low-dimensional process is smooth, we assume the hidden state evolves according to second-order Newtonian dynamics:

$$\mathbf{A} = \begin{bmatrix} 1 & A_v & 0 \\ 0 & 1 & A_a \\ 0 & 0 & 1 \end{bmatrix}, \tag{5.27}$$

$$C = [100]. \tag{5.28}$$

The components of the state have intuitive physical analogs: the first component corresponds to a position, the second to velocity, and the third to acceleration. This dynamics model allows for a smooth fall off in the covariance between time frames. The decay of covariances of various linear models is shown in Figure 5-3.

Each component of the output is assumed to evolve independently in time. Putting this all together, we may search for each component individually, greatly speeding up the performance of the algorithm.

5.4.2 Synthetic Results

We first demonstrate our algorithm on a synthetic 2D manifold embedded in $\mathcal{R}^3$. The neighborhood structure of this manifold is difficult to estimate from high-dimensional data, so traditional manifold learning techniques perform poorly on this data set. Taking into account the temporal coherence between data points and using user supervision alleviates these problems.

Figure 5-4(top-middle) shows an embedding of the 2D Markov process shown in

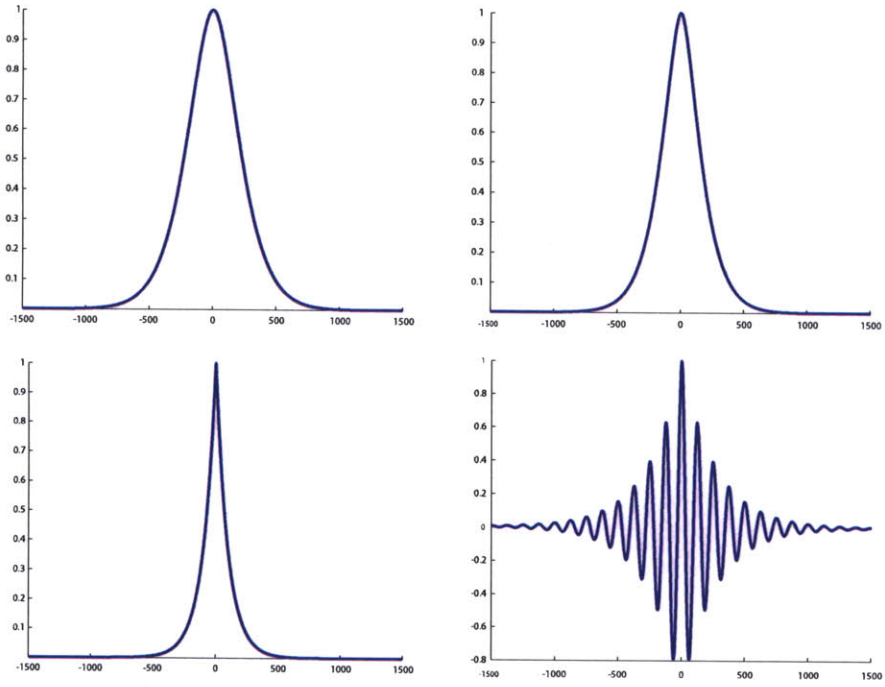

Figure 5-3: The covariance between samples over time for various $(\mathbf{A}, \mathbf{C})$ pairs. The x-axis represents number of samples from -1500 to 1500. The y-axis shows covariance on a relative scale from 0 to 1. (top-left) Newtonian dynamics model used in the experiments. (top-right) Dynamics model using zero acceleration. (bottom-left) Brownian Motion model. (bottom-right) A second order model with oscillatory modes.

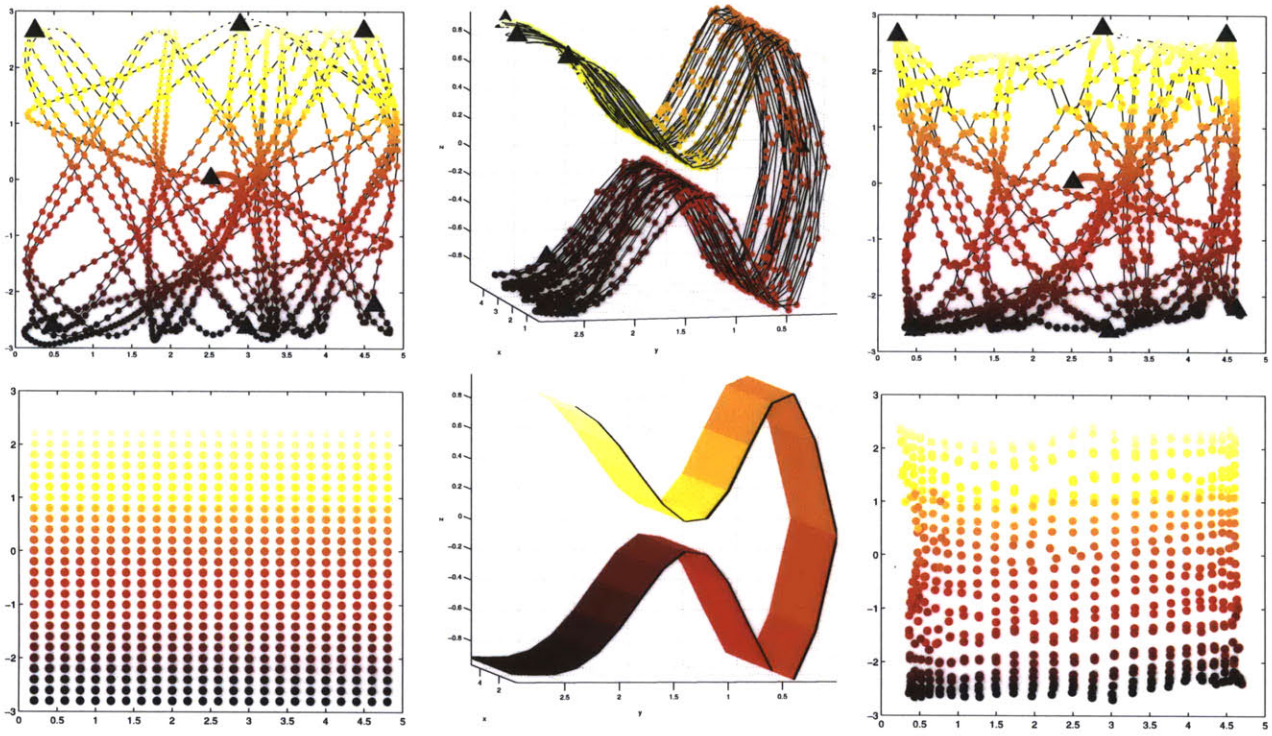

Figure 5-4: (top-left) The true 2D parameter trajectory. Semi-supervised points are marked with big black triangles. The trajectory is sampled at 1500 points (small markers). Points are colored according to their y-coordinate on the manifold. (top-middle) Embedding of a path via the lifting $F(x, y) = (x, |y|, \sin(\pi y)(y^2 + 1)^{-2} + 0.3y)$. (top-right) Recovered low-dimensional representation using our algorithm. The original data in (top-left) is correctly recovered. (bottom-left) Even sampling of the rectangle $[0, 5] \times [-3, 3]$. (bottom-middle) Lifting of this rectangle via F. (bottom-right) Projection of (bottom-middle) via the learned function g. g has correctly learned the mapping from 3D to 2D. These figures are best viewed in color.

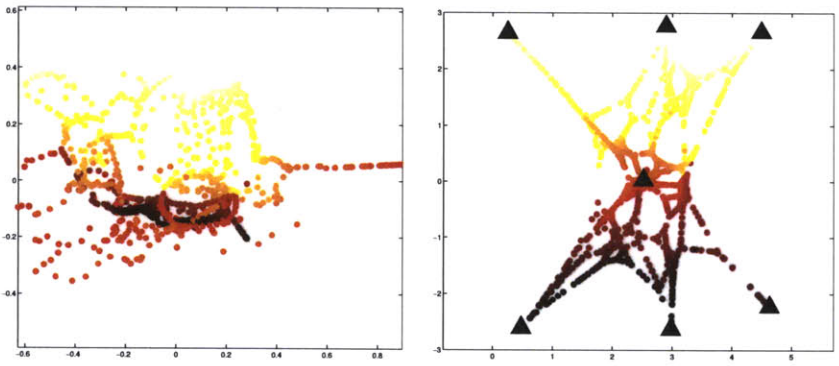

Figure 5-5: (left) Isomap's projection into $\mathcal{R}^2$ of the data set of Figure 5-4(top-middle). Errors in estimating the neighborhood relations at the neck of the manifold cause the projection to fold over itself. (right) Projection with BNR, a semi-supervised regression algorithm. There is no folding, but the projections are not close to the ground truth shown in Figure 5-4(top-left).

Figure 5-4(top-left) into $\mathcal{R}^3$. The semi-supervised points are marked with a large triangle. Figure 5-4(top-right) shows our interpolated results for the unlabelled points. The interpolated values are close to the true values that generated the data set. Although the process is smooth, it clearly does not follow the dynamics assumed by Equation (5.27) because it bounces off the boundaries of the rectangle $[0, 5] \times [-3, 3]$. Nevertheless, the assumed dynamics of Equation (5.27) are sufficient for recovering the true location of unlabelled points.

To assess the quality of the learned function g on as-yet unseen points, we evenly sampled the 2D rectangle $[0, 5] \times [-3, 3]$ and lifted the samples to $\mathcal{R}^3$ using the same mapping used to generate the training sequence. See Figure 5-4(bottom-left and bottom-right). Each sample in $\mathcal{R}^3$ is passed through g to obtain the 2D representation shown in Figure 5-4(bottom-right). The projections fall close to the true 2D location of these samples.

We applied LLE, Laplacian Eigenmaps, and Isomap to the data set of Figure 5-4(top-middle). Isomap produced the result shown in Figure 5-5(left). It is difficult to estimate the neighborhood structure near the neck, where the manifold comes close to intersecting itself, so Isomap creates folds in the projection.

Figure 5-5(right) shows the result of BNR. Compared to our result in Figure 5-4(top-right), the interpolated results are incorrect for most points. Since BNR does not attempt to enforce any geometric invariance in the projection, it is fairly robust to the neighborhood estimation problem.

For this and subsequent data sets, neither LLE nor Laplacian Eigenmaps produced sensible results. This may be due to the low rate at which the manifold is sampled.

5.4.3 Interactive Tracking

Our algorithm is not limited to rigid body tracking. We applied it to a lip tracking experiment exhibiting deformable motion, and to an upper-body tracking experiment exhibiting articulated motion. In these experiments, we restricted ourselves to recovering the missing labels of the training data and labeling frames acquired under the same setting from which the training data was gathered. Our algorithm op-

erates on the entire frames, as shown in the figures. Images were not in any way preprocessed before applying our algorithm, though to apply the learned mapping to different settings, more taylored representations or kernels could be employed. We tuned the parameters of our algorithm (A_v, A_a, the diagonal entries of Λ_ω, and the weights λ_d, λ_s, and λ_k) by minimizing the leave-one-out cross validation error on the semi-supervised points using the simplex method.

Figure 5-6 shows frames in a 2000 frame sequence of a subject articulating his lips. The top row shows the frames that were manually annotated with a bounding box around the lips. The bottom row shows the bounding boxes returned by g on some typical frames in the sequence. Only five labelled frames were necessary to obtain good lip tracking performance. The tracker is robust to natural changes in lighting, blinking, facial expressions, small movements of the head, and the appearance and disappearance of teeth.

Figure 5-7 shows 12 labelled images in a 2300 frame sequence of a subject moving his arms. These frames were manually labelled with line segments denoting the upper and lower arms. Figure 5-8 shows the recovered limb positions for unlabelled samples, some of which were not in the training sequence. Because the raw pixel representation is used, there are very few visual ambiguities between appearance and pose, and occlusions due to crossing arms do not present a problem.

The utility of dynamics is most apparent in articulated tracking. Setting λ_d to zero makes our algorithm ignore dynamics, forcing it to regress on the semi-supervised examples only. The resulting function produced the limb locations shown in black in Figure 5-8. Using dynamics allows the system to take advantage of the unsupervised points, producing better estimates of limb position.

5.4.4 Calibration of HCI Devices

The Audiopad is an interface for musical performance that aims to combine the modularity of knob based controllers with the expressive character of multidimensional tracking interfaces [75]. Audiopad uses a series of electromagnetically tracked objects, called pucks, as input devices. The performer assigns each puck to a set of samples

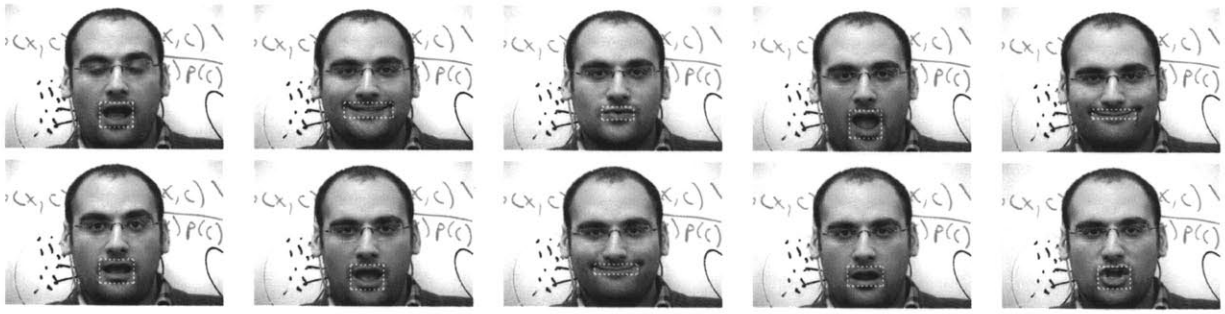

Figure 5-6: The bounding box of the mouth was annotated for 5 frames of a 2000 frame video. The labelled points (shown in the top row) and first 1500 frames were used to train our algorithm. The images were not altered in any way before computing the kernel. The parameters of the model were fit using leave-one-out cross validation on the labelled data points. Plotted in the second row are the recovered bounding boxes of the mouth for various frames. The first three examples correspond to unlabelled points in the training set. The tracker is robust to natural changes in lighting, blinking, facial expressions, small movements of the head, and the appearance and disappearance of teeth.

Figure 5-7: The twelve supervised points in the training set for articulated hand tracking (see Figure 5-8).

Figure 5-8: The hand and elbow positions were annotated for 12 frames of a 2300 frame video. The labelled points (shown in Figure 5-7) and the first 1500 frames were used to train our algorithm. The images were not preprocessed in any way. Plotted in white are the recovered positions of the hands and elbows. Plotted in black are the recovered positions when the algorithm is trained without taking advantage of dynamics. Using dynamics improves tracking significantly. The first two rows correspond to unlabelled points in the training set. The last row correspond to frames in the last 800 frames of the video, which was held out during training.

that he wishes to control. Audiopad determines the position and orientation of these objects on a tabletop surface and maps this data into musical cues such as volume and effects parameters. Graphical information is projected onto the tabletop surface from above, so that information corresponding to a particular physical object on the table appears directly on and around the object.

The Audiopad hardware is a result of further development of the Sensetable system [74]. The Sensetable tracks each puck using one or two RF tags. A simple type of RF tag, known as an LC tag, consists of a coil of wire and a capacitor. This circuit resonates at a specific frequency depending on its inductance and capacitance. Using clever antenna geometries, these simple structures can be tracked in space using amplitude measurements of the tags resonant frequencies. To determine the position of the RF tag on a two dimensional surface, a modified version of the sensing apparatus found in the Zowie^{TM}Ellie's Enchanted Garden^{TM}play set is used [21]. Each tag on the table resonates at a different frequency, so their positions can be determined independently, but there is a fair amount of black art in the decoding of the amplitude of resonances in the antenna to the tag-positions. As an exemplary embodiment, I will now describe how to learn the mapping from antenna resonance to tag position using semi-supervised regression with no knowledge of the particularities of the antenna geometry.

We wish to learn to map these 10 recorded measurments to the 2D position of the RFID tag. Previously, a mapping was recovered by hand through an arduous reverse-engineering process that involved building a physical model of the inner-workings of the Sensetable, and resorting to trial and error to refine the resulting mappings. Rather than reverse-engineering this device by hand, we show that it is possible to recover these mappings semi-automatically, with only 4 labelled examples and some unlabelled data points. This is a challenging task because the relationship between the tags position an the observed measurements is highly oscillatory. Once it is learned, we can use the mapping to track RFID tags. Of course, this procedure is quite general, and can be applied to a variety of other hardware.

To collect labelled examples, we placed the tag on each of the four corners of

the Sensetable and recorded the Sensetables output. We collected unlabelled data by sweeping the tag on the Sensetables surface for about 400 seconds, and down-sampled the result by a factor of 3 to obtain about 3600 unlabelled data points. The four labelled points, along with the few minutes of recorded data were passed to the semi-supervised learning algorithm to recover the mapping. Figure 5-10(left) shows the ground truth trajectory of the RFID tag, as recovered by the manually reverse-engineered Sensetable mappings. The four triangles in the corners of the figure depict the location of the labelled examples. The rest of the 2D trajectory was not made available to the algorithm. Figure 5-9(right) shows an example of the output from the Sensetable. Contrary to what one might hope, each trace of the output does not have a straightforward one-to-one relationship to a component of the 2D position. Rather, this relationship is smooth but sinusoidal. For example, when the tag is moved in a straight line from left to right, it generates sinusoidal traces similar to those shown in Figure 5-9(right).

The algorithm took 90 seconds to process this data set on a 2.8 Ghz Xeon machine. The trajectory is recovered accurately despite the complicated relationship between the 10 outputs and the tag position. See Figure 6. Its RMS distance to the ground truth trajectory is about 1.3 cm, though the ground truth itself is based on the reverse engineered tracker and may be inaccurate.

The mapping from measurements to positions is learned can be used to track tags. The recovered trajectories provide a subjective means of evaluating the accuracy of the tracker. Individual samples of 10 measurements can be passed to f to recover the corresponding tag position. Figure 5-11 shows the output of a few test paths. The recovered trajectories match the patterns traced by the tag.

The mapping cannot be learned from the four labelled examples alone using Tikhonov regularization, demonstrating that access to unlabelled data and prior knowledge about dynamics is very helpful in real-world applications. See Figure 5-12(left). Figure 5-12(middle) shows the trajectory recovered by BNR with its most favorable parameter setting for this data set. Figure 5-12(right) shows the trajectory recovered by BNR when temporally adjacent neighbors are counted as part of the ad-

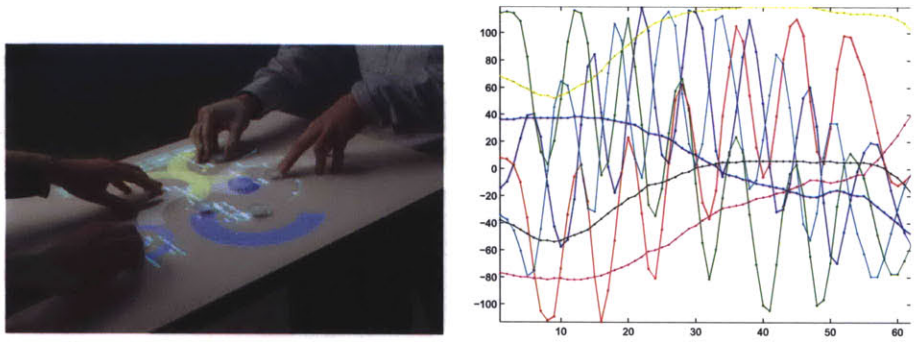

Figure 5-9: An image of the Audiopad. The plot shows an example stream of antenna resonance information. Samples from the output of the Sensetable over a six second period, taken over the trajectory marked by large circles in the left panel.

jacency graph when computing the Laplacian. As with the synthetic data set, there is severe shrinkage toward the mean of the labelled points, and some folding at the bottom. Taking temporal adjacency into account does not significantly improve the results.

5.4.5 Electric Field Imaging:

Anyone who has walked near a radio has likely noticed that the human body can cause interference with the transmission. This is because the human body acts as a capacitor coupling the radio frequencies to ground. This interference has been extensively utilized in the design of human computer interfaces and musical instruments at the Media Lab [33] [110] [72].

Some elementary field imaging sensor architectures have been developed by searching complex forward models [92]. A different approach was to combine several simple sensors [79]. Here I demonstrate a simple, inexpensive architecture based around a resistive sheet connected to a simple network of sub-dollar microcontrollers.

The Resistofish is a resistive sheet with electrodes along each side (see Figure 5-14). When a human is in the proximity of the sheet, it couples capacitively to the sheet. Based on the position of the body, there is a resistance between the coupling point and the electrodes. If one sensor is charged and another measures current with low impedance, the charging time can be used to estimate the time constant of the

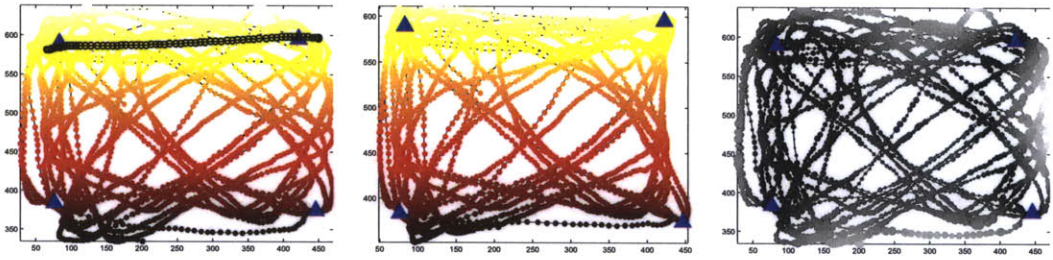

Figure 5-10: (left) The ground truth trajectory of the tag. The tag was moved around smoothly on the surface of the Sensetable for about 400 seconds, producing about 3600 signal strength measurement samples after downsampling. Triangles indicate the four locations where the true location of the tag was provided to the algorithm. The color of each point is based on its y-value, with higher intensities corresponding to higher y-values. (right) (middle) The recovered tag positions match the original trajectory. (right) Errors in recovering the ground truth trajectory. Circles depict ground truth locations, with the intensity and size of each circle proportional to the Euclidean distance between a points true position and its recovered position. The largest errors are outside the bounding box of the labelled points. Points in the center are recovered accurately, despite the lack of labelled points there.

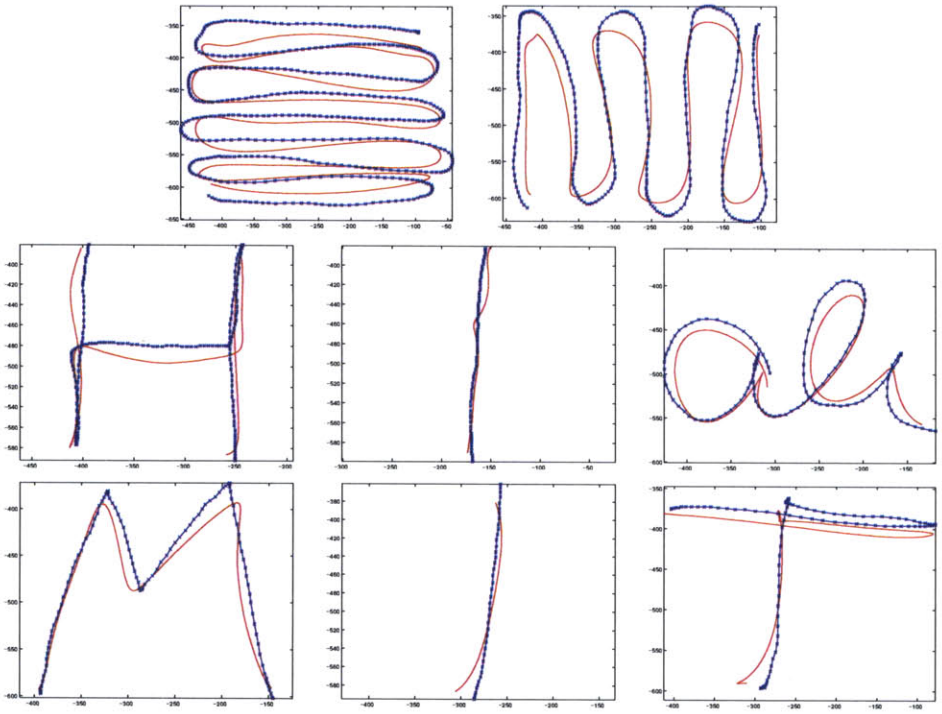

Figure 5-11: Once f is learned, it can be used it to track tags. Each panel shows a ground truth trajectory (blue crosses) and the estimated trajectory (red dots). The recovered trajectories match the intended shapes.

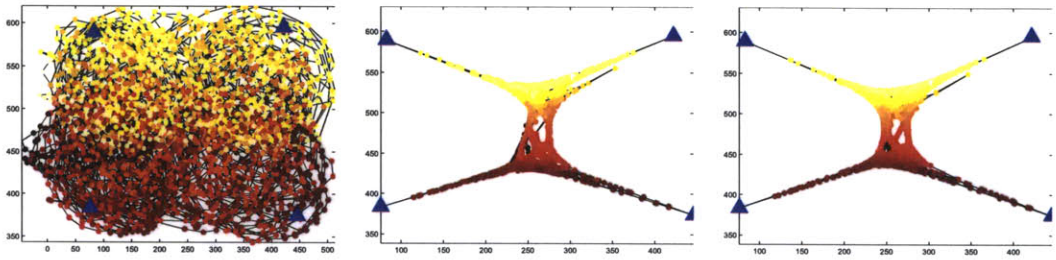

Figure 5-12: (left) Tikhonov regularization with labelled examples only. The trajectory is not recovered. (middle) BNR with a neighborhood size of three using nearest neighbors. (right) BNR with same neighborhood settings, with the addition of temporal neighbors. There is folding at the bottom of the plot, where black points appear under the red points, and severe shrinking towards the mean.

RC pair.

Using the four electrode pairs, we trained two hand trackers with the resistive sheets. The first was made by creating a 7×12 grid on the sheet and measuring approximately 30 samples per point. We then used leave-one-out cross validation to train a kernel machine to map sensor outputs to hand positions. The second tracker was made using the unsupervised regression algorithm with a dynamics prior. Creating a similar 2D tracing, we learned the best mapping, with no parameter tuning and 200 samples. The results of a test tracing are displayed in Figure 5-15.

Both algorithms recovered geometry to similar accuracy (see Figure 5-16), but the supervised algorithm was far more laborious to train. While the unsupervised method only took 30 seconds to acquire the data, and 10 seconds to process it, the supervised algorithm took nearly an hour to gather the data, and an hour to perform the cross validation.

5.5 Conclusion

We have presented a semi-supervised regression algorithm for learning the appearance manifold of a scene from a video sequence. By taking advantage of the dynamics in video sequences, our algorithm learns a function that projects images to a low-dimensional space with semantically meaningful coordinate axes. The experiments

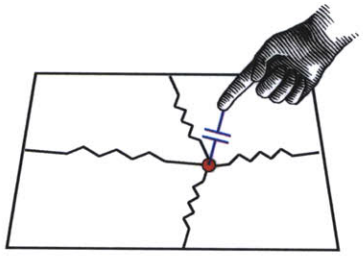

Figure 5-13: The Resistofish senses humans by detecting the low-level electric fields that couple them to ground. The hand couples capacitively to a resistive sheet with electrodes on the sides. The time constant of the RC pair that couple the hand to the sheet are measured by undersampling timing the impulse response of a voltage change at each electrode.

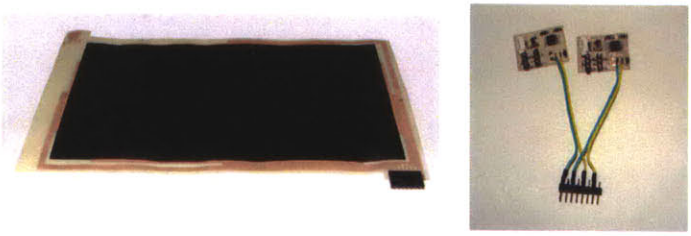

Figure 5-14: The resistive sheet and the two dollar sensor that make up the Resistofish hardware.

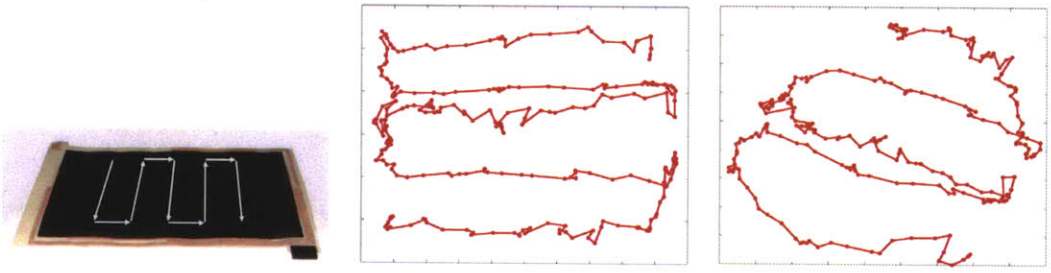

Figure 5-15: Two different algorithms were used to measure the mapping from the RC time constants to the position of the hand. (left) A sample trajectory. (middle) The recovered trajectory under the supervised algorithm. (left) The recovered trajectory by the unsupervised regression algorithm. Note that the trajectory is rotated, but the geometry is correctly recovered.

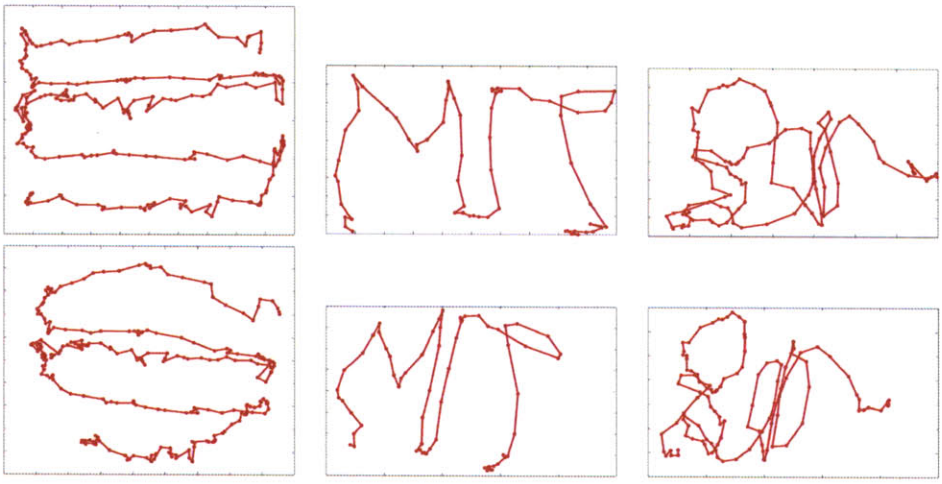

Figure 5-16: The top row is recovered using the supervised algorithm. The bottom row is recovered by the unsupervised algorithm. The middle panels is the recovered traces of someone writing "MIT." The right-most panels are the recovered traces of someone writing "Ben." The mapping recovered by the unsupervised algorithm is as useful for tracking human interaction as the mapping recovered by the fully calibrated regression algorithm.

demonstrate that this optimization framework is a powerful way to build trackers with very little domain specific knowledge and very few data points.

Chapter 6

Output Prior: Manifolds of Low Codimension

In this final chapter, I will describe a novel algorithm for learning manifolds defined by observed data. Most algorithms that claim to learn manifolds can only operate under restrictive locality assumptions, and, even more problematically, can only learn mappings that are 1-1 [97, 85, 8, 27, 106, 16]. The most simple example of a manifold, the sphere, breaks all of the present algorithms.

Here, I take a different approach. I will learn functions on the high dimensional space that are constant on the manifold. These functions will define the manifold structure, with the space normal to the manifold spanned by the gradients of these functions.

This method is similar to the novelty deteciton algorithms that use kernels to generate inequality constraints that bound the data [96, 10, 88]. Here, we will learn equality constraints, and in turn, implicitly defined manifolds on which the data lie.

6.1 Learning Manifolds of Low Codimension

Let $\mathcal{M}$ be a surface in embedded in $\mathbf{R}^d$ implicitly defined by unknown differentiable functions $f_1, \ldots, f_J$. That is,

$$\mathcal{M} = \{\mathbf{x} \in \mathbb{R}^d \,:\, f_j(\mathbf{x}) = 0 \quad j = 1, \ldots J\} \tag{6.1}$$

Let us further assume that $\{Df_j(\mathbf{x})\}$ span a J-dimensional space for every $\mathbf{x} \in \mathcal{M}$. Such a surface is a manifold of codimension J by the implicit function theorem. Let $\mathbf{x}_i$ be a collection of N points sampled from $\mathcal{M}$ via some unknown probability distribution p on $\mathcal{M}$.

We assume that both p and $f_1, \ldots, f_J$ are unknown. Let us even assume the codimension J is unknown. The goal will be to approximate the unknown functions f_j from the data $\mathbf{x}_i$

(i) Lift the data to a higher dimensional Hilbert space $\mathcal{H}$ via a mapping $\Phi : \mathbf{R}^d \to \mathcal{H}$.

(ii) Using linear algebra, find a set of vectors $\mathbf{a}_j$, $j = 1, \ldots, k$ such that $\mathbf{a}_j^\top \Phi(\mathbf{x}_i) = \mathbf{0}$ for all i.

(iii) Return $f_j(\mathbf{x}) := \mathbf{a}_j^\top \Phi(\mathbf{x})$

The only difficulty is in choosing the mapping Φ so that the output function is stable and produces reasonable models. Following the theme of the thesis, we will choose f_j to live in a Reproducing Kernel Hilbert Space. There are two cases we will investigate. First, we will discuss the case of finite dimensional liftings and, in particular, consider the case when f_j are polynomials. Then we will focus on the generic case where f_j live in an arbitrary RKHS.

What is the advantage of learning multiple functions which are zero on the data set? And how many functions are needed to define an implicit surface? The answer comes from a variant of the implicit function theorem which states that if $g : \mathbb{R}^d \to \mathbb{R}^J$ with $J \leq d$, then if if Dg has rank J at every $\mathbf{x}$ with $g(\mathbf{x}) = 0$, then the preimage of

0 under g is a manifold of dimension $d - J$ [41]. J is called the *codimension* of the manifold. It is the codimension of the tangent space of the manifold in $\mathbb{R}^d$ for every point $\mathbf{x} \in \mathcal{M}$. The co-dimension is stable to small perturbations of the map g. That is, Dg remains full rank even if g is perturbed by a small amount, and, in turn, the preimage of 0 remains a $d - J$-dimensional manifold.

Let us now consider the situation where we have learned L functions g_l which are all identically zero on $\mathcal{M}$. If $L < J$ then the dimension of the zero-set of the g_l is of larger dimension than that of $\mathcal{M}$. If on the other hand, $L > J$, we know that the rank of Dg is at most J. In this case, we can use the rank of Dg to estimate the codimension of the manifold and from this we can try to learn a set of J functions which are zero only on $\mathcal{M}$. In what follows, we will learn $d + 1$ functions, more than could ever be necessary to define $\mathcal{M}$. From these functions, we will estimate both the codimension of $\mathcal{M}$ and the smallest set of functions necessary to describe $\mathcal{M}$.

6.2 Basis Functions and Polynomial Models

The first option to consider is to choose a set of differentiable basis functions b_α for $L_2(\mathbb{R}^d)$ and lift $\mathbf{x}$ to a subset of those functions

$$\Phi(\mathbf{x}) := \begin{bmatrix} b_1(\mathbf{x}) & \cdots & b_T(\mathbf{x}) \end{bmatrix}^\top \tag{6.2}$$

To compute a constraint, compute the SVD of the matrix $\mathbf{A} = [\Phi(\mathbf{x}_1), \ldots, \Phi(\mathbf{x}_N)]$ If $\mathbf{A}$ does not have full row rank, than any vector in the left null space will suffice for the desired vector $\mathbf{a}$.

For example, if we wanted the constraint to be algebraic, we could lift to a vector of monomials of bounded degree

$$\Phi\left(\begin{bmatrix} x_1 & \cdots & x_d \end{bmatrix}^\top\right) := \begin{bmatrix} 1 & x_1 & \cdots & x_d & x_1x_2 & x_1x_3 & \cdots \end{bmatrix}^\top. \tag{6.3}$$

The utility of such an approach can be illustrated by supposing the data was drawn

from the unit sphere in $\mathbb{R}^3$. If we lift the data as

$$\Phi\left(\begin{bmatrix} x_1 & x_2 & x_3 \end{bmatrix}^\top\right) := \begin{bmatrix} 1 & x_1 & x_2 & x_3 & x_1^2 & x_2^2 & x_3^2 & x_1x_2 & x_2x_3 & x_1x_3 \end{bmatrix}^\top \quad (6.4)$$

one would find the vector

$$\mathbf{a}^\top = \begin{bmatrix} 1/2 & 0 & 0 & 0 & -1/2 & -1/2 & -1/2 & 0 & 0 & 0 \end{bmatrix}^\top \quad (6.5)$$

annihilates all points of the lifting. And this, of course, is the constraint $x_1^2+x_2^2+x_3^2 = 1$.

6.3 Lifting to a General RKHS

Suppose we wish to search for a radial basis function of the form

$$f(\mathbf{x}) = \sum_i c_i k(\mathbf{x}_i, \mathbf{x}) \quad (6.6)$$

where $k(\mathbf{x}, \mathbf{y})$ is a positive definite kernel. Unfortunately, such an expression cannot define the surface $\mathcal{M}$.

Proposition 6.3.1 *Suppose* $f(\mathbf{x}) = \sum_i c_i k(\mathbf{x}_i, \mathbf{x})$. *Then* $f(\mathbf{x}_j) = 0$ *for all* j *if and only if* $f(\mathbf{x}) \equiv 0$.

Proof Since $f(\mathbf{x}) = \langle f, k(\mathbf{x}, \cdot)\rangle_K$ in the RKHS,

$$0 = f(\mathbf{x}_j) = \langle f, k(\mathbf{x}_i, \cdot)\rangle_K = \langle \sum_i c_i k(\mathbf{x}_i, \cdot), k(\mathbf{x}_j, \cdot)\rangle_K \quad (6.7)$$

implies that f is a vector in the span of $k(\mathbf{x}_i, \cdot)$ which is orthogonal to all of the $k(\mathbf{x}, \cdot)$ which implies that $f = 0$. ■

We can conclude from this proposition any non-zero function in the RKHS which is zero on $\mathcal{M}$ must be orthogonal to $k(\mathbf{x}, \cdot)$ for all $\mathbf{x} \in \mathcal{M}$. This inspires the following approach: let $\alpha_1, \ldots, \alpha_s$ be a basis for the span of $\{k(x_1, \cdot), \ldots, k(x_N, \cdot)\}$. If we pick

some basis functions $b_1, \ldots, b_t$ in the RKHS which are not contained in the span of the data, we can perform Gram-Schmidt on the set $\{b_1, \ldots, b_t, \alpha_1, \ldots, \alpha_s\}$ to yield t functions $g_1, \ldots, g_t$ which are orthonormal in the RKHS and are orthogonal to the span of the data. We would then have $g_j(\mathbf{x}_i) = 0$ for all i. And, in the limit of enough data, we would have $g_j(\mathbf{x}) = 0$ for all $\mathbf{x} \in \mathcal{M}$.

In practice, this amounts to fixing the expansion for g_j as

$$g_j(\mathbf{x}) = \sum_{m=1}^{t} a_{jm} b_m(\mathbf{x}) + \sum_{n=1}^{N} c_{jn} k(\mathbf{x}_n, \mathbf{x}) \tag{6.8}$$

and searching for $[\mathbf{a}, \mathbf{c}]$ such that $g_j(\mathbf{x}) = 0$. That is, searching in the null space of the rows of

$$\mathbf{A} = [\Phi(\mathbf{x}_1), \ldots, \Phi(\mathbf{x}_N)] \tag{6.9}$$

where

$$\Phi(\mathbf{x}) := \begin{bmatrix} b_1(\mathbf{x}) & \cdots & b_t(\mathbf{x}) & k(\mathbf{x}_1, \mathbf{x}) & \cdots & k(\mathbf{x}_N, \mathbf{x}) \end{bmatrix}^\top . \tag{6.10}$$

There are thus two different interpretations of the lifting when using radial basis functions. The first is that we have chosen a set of basis functions based on the data $\mathbf{x}_1, \ldots, \mathbf{x}_N$ and then lifted to this basis set as in Section 6.2. The second interpretation is that we are lifting each data point to the infinite dimensional RKHS under the mapping $\mathbf{x} \mapsto k(\mathbf{x}, \cdot)$ and are searching for an element in the RKHS orthogonal to the data under this lifting.

6.4 Null Spaces and Learning Surfaces

Note that we could have tried to learn a set of functions g_j such that

$$g_j(\mathbf{x}_i) = g_j(\mathbf{x}_k) \quad \forall i, k \tag{6.11}$$

This function would be *constant* on the data set. A function that is zero on the data set could be produced by setting $\hat{g}_j = g_j - g_j(\mathbf{x}_1)$.

Define the extended kernel matrix to be the $N \times (N+t)$ matrix $\hat{\mathbf{K}} = [\mathbf{K}, \mathbf{B}]$ where $\mathbf{B}$ has entries

$$B_{ij} = b_j(\mathbf{x}_i) \tag{6.12}$$

For each M, define the $(M-1) \times M$ matrix $\mathbf{A}[M]$ to be

$$A[M]_{ij} = \begin{cases} 1 & 1 \leq i = j < M-1 \\ -1 & 1 \leq i = j-1 < M-1 \\ 0 & \text{otherwise} \end{cases} \tag{6.13}$$

that is

$$\mathbf{A}[M] = \begin{bmatrix} 1 & -1 & 0 & 0 & \dots & 0 \\ 0 & 1 & -1 & 0 & \dots & 0 \\ & & \ddots & & & \\ 0 & \dots & 0 & 1 & -1 & 0 \\ 0 & \dots & 0 & 0 & 1 & -1 \end{bmatrix} \tag{6.14}$$

A function of the form (6.16) satisfies (6.15) for all i and k between 1 and N if and only if $\mathbf{A}[N]\hat{\mathbf{K}}[\mathbf{c}, \mathbf{a}]^\top = 0$. In particular, the coefficients of the expansion in (6.16) lie in the null space of $\mathbf{A}[N]\hat{\mathbf{K}}$. In this case, we can find the space of all functions that are constant on the data set.

This approach has a nice generalization to the case where we are searching for a function that is constant on multiple data sets. For example, if we are given multiple trajectories of a conservative system, each trajectory will have constant energy, but two different trajectories will likely have different energy.

In this case, we are presented with data partitioned into disjoint subsets $S_k \subset \{1, \dots, N\}$. We seek to find g_j such that

$$\mathbf{g}_j(\mathbf{x}_m) = \mathbf{g}_j(\mathbf{x}_n) \quad \text{if } m, n \in S_k \text{ for some } k \tag{6.15}$$

Assume without loss of generality, that each S_k is a contiguous subset of $\{1, \dots, N\}$ of size N_k. In this case, we may define the matrix $\mathbf{A} = \text{diag}(\mathbf{A}[N_1], \dots, \mathbf{A}[N_k])$ and

search for coefficients in the null space of $\mathbf{A}\hat{\mathbf{K}}$.

6.5 Choosing a Basis

Once a desired space of functions is learned, a basis can be selected by finding a set of coefficients, $\mathbf{C}$, such that, the functions $\mathbf{C}$ are constant on the data, and the gradients of the functions $\mathbf{C}$

$$\nabla g_j(\mathbf{x}_i) = \sum_{m=1}^{t} a_{jm} \nabla b_m(\mathbf{x}_i) + \sum_{n=1}^{N} c_{jn} \nabla k(\mathbf{x}_n, \mathbf{x}_i) \tag{6.16}$$

are linearly independent for each i. Since this condition is linear in the coefficients, we can jointly search for $\mathbf{C}$ that define functions that are constant on the data and whose span is linearly independent.

6.6 Learning Manifolds

I have performed experiments with several data sets in $\mathbb{R}^3$. In all cases, I used a gaussian kernel as in Equation (2.74) with $C = 1$. The basis functions were chosen to be the constant function and the three linear functions x, y, and z. In Figures 6-1 and 6-2 I show the results on fitting the 2-sphere. I generated 200 uniformly distributed points shown in Figure 6-1 by sampling from a gaussian and then normalizing the vectors to have length 1. In Figure 6-2 I show the recovered zero-sets for the four functions, with the rightmost frame showing the intersection of the four zero sets. Similar results were obtained for the confectionary data-sets DOUGHNUT and SWISS and shown in Figures 6-3, 6-4, 6-5, and 6-6. In all cases, the dimension of the space spanned by the gradients of the functions which defined the zero-set were correctly estimated to be 1, the codimension of all of the manifolds.

The learned functions are robust to noise in the data. We can demonstrate this by showing $|g_j(\mathbf{x})| < \epsilon$ on the true manifold. On the SPHERE data set, adding gaussian noise with covariance $10^{-3}\mathbb{1}$ results in a function which is less that $10^{-4}M$ on the

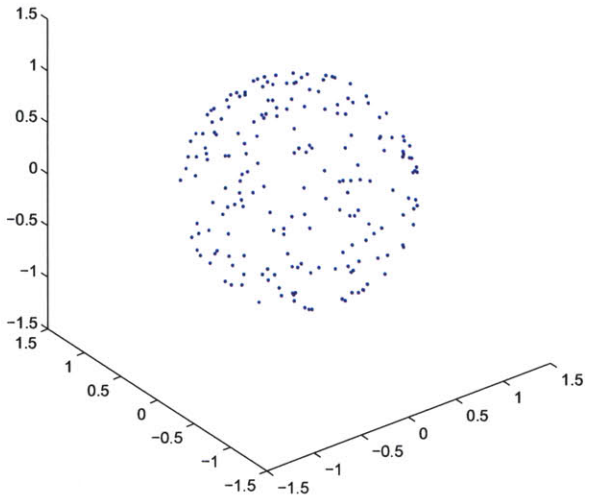

Figure 6-1: The SPHERE data set. 200 points were sampled from a gaussian with unit variance and then normalized to have length 1. This sampling procedure generates a uniform distribution on the sphere.

entire sphere where

$$\begin{aligned} M = \quad & \max_{j,\mathbf{x}} \quad g_j(\mathbf{x}) \\ & \text{s.t.} \qquad \mathbf{x} \in [-2,2] \times [-2,2] \times [-2,2] \end{aligned} \tag{6.17}$$

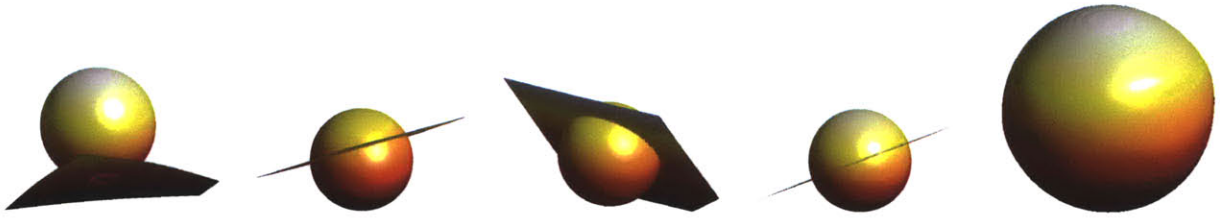

Figure 6-2: The first four figures show the zero-contours of four functions whose coefficients span the null-space of lifted data for SPHERE. The final figure shows the intersection of these four surfaces. This plot is computed by calculating the zero contour of the sum of the squares of the four functions.

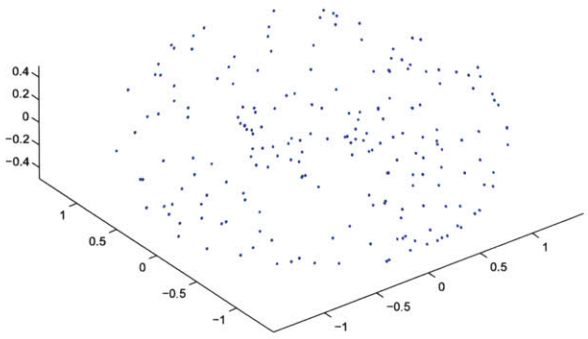

Figure 6-3: The DOUGHNUT data set. 200 points were sampled uniformly from the box $[0, 2\pi] \times [0, 2\pi]$ and then lifted by the map $(x, y) \mapsto (\cos(x) + \frac{1}{2}\cos(y)\cos(x), \sin(x) + \frac{1}{2}\cos(y)\sin(x), \frac{1}{2}\sin(y))$

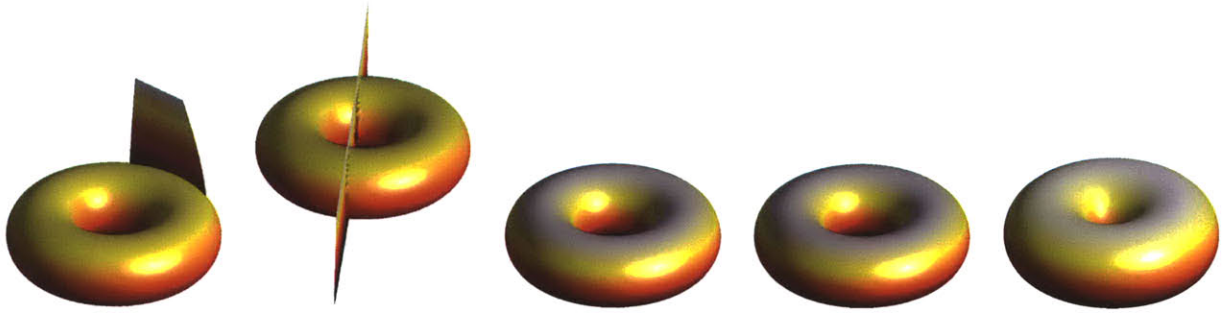

Figure 6-4: The first four figures show the zero-contours of four functions whose coefficients span the null-space of lifted data for DOUGHNUT. The final figure shows the intersection of these four surfaces. This plot is computed by calculating the zero contour of the sum of the squares of the four functions.

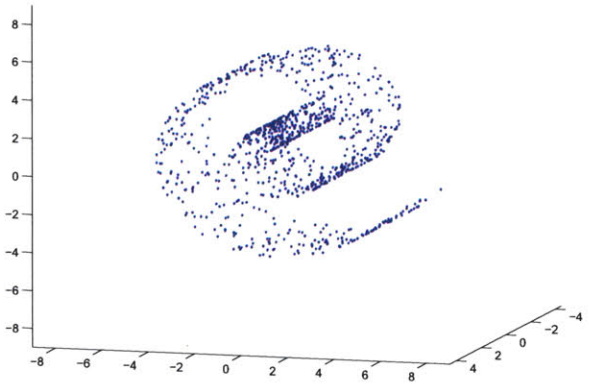

Figure 6-5: The SWISS data set. 1000 points were sampled uniformly from the box $[0, 5] \times [0, 6]$ and then mapped $(x, y) \mapsto (x, |y| \cos(2y), |y| \cos(2y))$.

Figure 6-6: The first four figures show the zero-contours of four functions whose coefficients span the null-space of lifted data for SWISS. The final figure shows the intersection of these four surfaces. This plot is computed by calculating the zero contour of the sum of the squares of the four functions.

Chapter 7

Conclusion

Beginning with one simple cost function, I have presented a powerful framework for semi-supervised and unsupervised learning. The optimizations defined by this cost function can be principally approximated using Lagrangian duality. Furthermore, these approximations can be optimized efficiently using standard algorithms. In this way, once we have posed a learning problem in our framework, its solution is immediate. This puts the burden on the practitioner to pose the problem, not to slave over creating novel algorithms.

The utility of this framework was demonstrated through four very different applications. In each case, I adjoined a simple and intuitive prior to the function learning problem and then showed how such simple additions immediately produced powerful algorithms. First, I considered priors on the kernel function that generates the RKHS. From such priors, new insights into Kernel Learning were derived, and a novel method for learning polynomial kernels was presented. Second, I showed how the prior of binary labels lead to powerful algorithms for transductive classification and segmentation. The segmentation algorithm generalizes the so-called spectral clustering algorithms. Third, I imposed a dynamics prior on the labels and developed an optimization that can learn a mapping between two time series. This optimization takes as input examples of how to map individual samples of the input time series to corresponding samples of the output time series. Because it assumes that the output time series follows known dynamics, it can also take advantage of unlabelled input

samples. Finally, by applying the prior that the hidden function was constant on the data, a new method for learning manifolds based on implicit functions was derived.

There are a variety of directions that remain to be explored. With a particular focus on the examples presented in this thesis, there are many interesting open problems. For clustering, it would be interesting to develop methods for multiclass classification that do not require successive application of two-way clustering algorithms. It would be useful to devise a method that, when given an initial guess at the number of clusters, K, tries to find the best K-way partitioning at once. For the dynamics algorithms, the state transition matrices were fixed *a priori*. Applying techniques from system identification [60, 71, 102], it may be possible to learn these as well by modifying the current optimization. Furthermore, a method for semi-supervised regression with a nonlinear dynamics model would be of great interest. The techniques developed in Chapter 6 could be used to empirically study the conserved quantities of dynamical systems [3, 59]. Furthermore, such methods could be used for nonlinear system identification by finding the manifolds that are constant on the trajectories of dynamical systems. There is an endless variety of domain specific applications that can be derived with new output priors, and it would be interesting to expand the toolbox of techniques that fall under this framework.

I am particularly interested in exploring further applications of the duality tools used in Chapter 3. Recall that I was able to construct a proof of the Representer Theorem using duality on the Hilbert Space of functions. Inspired by the polynomial relaxation techniques that exploit positivity conditions in the dual problems [58, 73], I hope to develop new techniques for solving quadratic problems where the primal problem deals with very high dimensional objects. Such relaxations provide finite dimensional convex programs that operate on sufficient statistics of the original optimization program. For example, in Chapter 3, I only needed to consider the inner products of the infinite dimensional f with a finite set of vectors. It remains to be seen if we can generalize past the results of Chapter 3 to provide further insights into the structure of Reproducing Kernel Hilbert Spaces and devise new techniques in machine learning.

There are two other areas where I believe such techniques can yield powerful new algorithms. First, there has been substantial activity in the controls community to develop analysis and synthesis tools for systems consisting of extremely large numbers of interconnected subsystems. A large part of this effort has been devoted to developing tools that scale gracefully with the number of subsystems, and their attendant local sensing, actuating, and computing elements. Clearly for systems that are comprised of a large number of subsystems, structure must be fully exploited to obtain tractable analysis and control synthesis algorithms.

Recent work has made a great deal of progress in exploiting the symmetry present in such systems. Control laws can be distributed such that they only rely on local communication, yet can still give rise to desired global behavior, and, in certain settings, it has been shown that spatially distributed controllers are optimal for the control of spatially invariant systems [5, 65]. The synthesis of such distributed controllers is often convex [26, 104], and taking the distributed structure of a problem into account can greatly reduce the complexity of control design without sacrificing system performance [18].

I want to connect the recently presented techniques for the control design of spatially interconnected systems [22, 23, 24] to the relaxation of duality on Hilbert spaces. I have shown how these results are in fact applicable to a much larger class of interconnection topologies where the symmetry of the interconnection may be noncommutative [81, 82]. The dual techniques developed in this thesis may provide further analysis conditions which guarantee performance objectives for these interconnected systems.

Secondly, I want to explore applications in quantum information. For a quantum system, the *Hamiltonian* is the mathematical object which both governs the evolution of the system and the energies of the various configurations. A fundamental problem is to determine the configuration with the lowest possible energy. Such a configuration is called the *ground state* of a given Hamiltonian and its corresponding energy level is called the *ground state energy* [57].

The ground state energy is generally NP-HARD to calculate. Even the simple

looking problem of configuring an array of small quantum magnets, called spins, to minimize their interaction energy is NP-HARD [6], although special cases can be solved analytically [54]. Furthermore, it often requires exponentially many numbers to describe the ground state itself. Since it is likely impossible to exactly calculate the ground state or ground state energy of a generic quantum system, alternative approximate means are needed to gain insight into the structure of the quantum ground state. Many approximate techniques have been developed. Early techniques used perturbations around a so-called "mean field" to generate useful estimates of molecular structure [95]. The Density Matrix Renormalization Group [107] has been widely successful in the study of chains of identical systems, but does not scale to arbitrarily coupled topologies. Chemists in the 1950s developed analytic tools for studying electrons [64, 66] that could only be solved in simple cases until they were shown half a century later to be solvable by semidefinite programming [67].

I intend to study a new approach to estimating the ground state energy using approximations of the dual problem. Noting that most Hamiltonians are presented as sums of products of local interaction terms, redundant constraints amongst these interactions can be adjoined to the formulation of the ground state problem. These interactions may be easy to analyze on their own and their energy levels can be added as redundant constraints. By relaxing the dual, I hope to produce optimization problems that are polynomial in the number of interactions. I am particularly interested in exploring how these techniques extend the work on electrons [67] to a more widely applicable framework and how this new approach might generalize the algorithm to the Goemans-Williamson algorithm for MAX-CUT [37] to the quantum regime.

Appendix A

Linear Algebra

Many of the algebraic manipulations used in this document are based on techniques derived from minimizing quadratic forms. Here we present some of these derivations.

A.1 Unconstrained Quadratic Programming

Given a matrix $\mathbf{A}$, one can readily check that

$$\min_{\mathbf{x}} \mathbf{x}^\top \mathbf{A}\mathbf{x} = \begin{cases} 0 & \mathbf{A} \succeq 0 \\ -\infty & \text{otherwise} \end{cases} \tag{A.1}$$

This is because if $\mathbf{A}$ has any negative eigenvalues, then the cost is unbounded below. Assume $\mathbf{A}$ is positive semidefinite and consider the quadratic program

$$\min_{\mathbf{x}} \mathbf{x}^\top \mathbf{A}\mathbf{x} - 2\mathbf{b}^\top \mathbf{x} + c \tag{A.2}$$

we may, differentiate with respect to $\mathbf{x}$ to find that at the optimum

$$\mathbf{A}\mathbf{x} = \mathbf{b} \tag{A.3}$$

If $\mathbf{A}$ is invertible then the minimum is $-\mathbf{b}^\top \mathbf{A}^{-1}\mathbf{b} + c$

A.2 Schur Complements

Let $\mathbf{M}$ be a matrix partitioned as

$$\mathbf{M} = \begin{bmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{B}^\top & \mathbf{C} \end{bmatrix} \tag{A.4}$$

For the simplicity of presentation, let us assume that $\mathbf{A}$ and $\mathbf{B}$ are invertible. The Schur complement of $\mathbf{A}$ in $\mathbf{M}$ is defined to be

$$(\mathbf{M}|\mathbf{A}) = \mathbf{C} - \mathbf{B}^\top \mathbf{A}^{-1} \mathbf{B} \tag{A.5}$$

and the Schur complement of $\mathbf{C}$ in $\mathbf{M}$ is

$$(\mathbf{M}|\mathbf{C}) = \mathbf{A} - \mathbf{B}\mathbf{C}^{-1}\mathbf{B}^\top \tag{A.6}$$

A.3 More Quadratic Programming

For the quadratic minimization

$$\min_{\mathbf{x}_2} \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \end{bmatrix}^\top \begin{bmatrix} \mathbf{A} & \mathbf{B}^\top \\ \mathbf{B} & \mathbf{C} \end{bmatrix} \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \end{bmatrix} - 2 \begin{bmatrix} \mathbf{b}_1 \\ \mathbf{b}_2 \end{bmatrix}^\top \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \end{bmatrix} \tag{A.7}$$

the optimal $\mathbf{x}_2^*$ is given by

$$\mathbf{x}_2^* = \mathbf{C}^{-1}(\mathbf{b}_2 - \mathbf{B}\mathbf{x}_1) \tag{A.8}$$

Plug that back into the cost function:

$$\mathbf{x}_1^\top (\mathbf{M}|\mathbf{C})\mathbf{x}_1 - 2(\mathbf{b}_1 - \mathbf{B}\mathbf{C}^{-1}\mathbf{b}_2)^\top \mathbf{x}_1 \tag{A.9}$$

Similarly, minimizing over $\mathbf{x}_1$ gives the cost function

$$\mathbf{x}_2^\top (\mathbf{M}|\mathbf{A})\mathbf{x}_2 - 2(\mathbf{b}_2 - \mathbf{B}^\top \mathbf{A}^{-1}\mathbf{b}_1)^\top \mathbf{x}_2 \tag{A.10}$$

Solving these two reduced quadratic programs gives

$$\begin{bmatrix} \mathbf{x}_1^* \\ \mathbf{x}_2^* \end{bmatrix} = \begin{bmatrix} (\mathbf{M}|\mathbf{C})^{-1}(\mathbf{b}_1 - \mathbf{B}\mathbf{C}^{-1}\mathbf{b}_2) \\ (\mathbf{M}|\mathbf{A})^{-1}(\mathbf{b}_2 - \mathbf{B}^\top\mathbf{A}^{-1}\mathbf{b}_2) \end{bmatrix} \tag{A.11}$$

A.4 Inverting Partitioned Matrices

Since

$$\begin{bmatrix} \mathbf{x}_1^* \\ \mathbf{x}_2^* \end{bmatrix} = \mathbf{M}^{-1} \begin{bmatrix} \mathbf{b}_1 \\ \mathbf{b}_2 \end{bmatrix} \tag{A.12}$$

we see that

$$\mathbf{M}^{-1} = \begin{bmatrix} (\mathbf{M}|\mathbf{C})^{-1} & -(\mathbf{M}|\mathbf{C})^{-1}\mathbf{B}\mathbf{C}^{-1} \\ -(\mathbf{M}|\mathbf{A})^{-1}\mathbf{B}^\top\mathbf{A}^{-1} & (\mathbf{M}|\mathbf{A})^{-1} \end{bmatrix} \tag{A.13}$$

A.5 Schur complement Lemma

One of the most important tools used throughout the document is the Schur Complement Lemma. This Lemma often allows for the transformation of polynomial expressions into semidefinite constraints that are linear in the parameters of interest.

Lemma A.5.1 (The Schur Complement Lemma)

$$\mathbf{M} \succeq 0 \iff \mathbf{C} \succeq 0 \quad \textit{and} \quad (\mathbf{M}|\mathbf{C}) \succeq 0 \tag{A.14}$$

Proof The ($\Longrightarrow$) direction is true because subblocks of positive semidefinite matrices are positive semidefinite and $\mathbf{M}^{-1}$ is positive semidefinite when $\mathbf{M}$ is.

To prove the reverse direction, consider minimizing (A.7) with $\mathbf{b}_1 = \mathbf{b}_2 = 0$. Then for any $\mathbf{x}$,

$$\min_{\mathbf{x}_2} \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \end{bmatrix}^\top \mathbf{M} \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \end{bmatrix} = \mathbf{x}_1^\top(\mathbf{M}|\mathbf{C})\mathbf{x}_1 \geq 0 \tag{A.15}$$

and hence $\min_{\mathbf{x}} \mathbf{x}^\top\mathbf{M}\mathbf{x} \geq 0$ and $\mathbf{M}$ is positive semidefinite. ■

A.6 Matrix Inversion Lemma

$$(\mathbf{A} - \mathbf{B}\mathbf{C}^{-1}\mathbf{B}^\top)^{-1} = \mathbf{A}^{-1} + \mathbf{A}^{-1}\mathbf{B}(\mathbf{C} - \mathbf{B}^\top\mathbf{A}^{-1}\mathbf{B})^{-1}\mathbf{B}^\top\mathbf{A}^{-1} \tag{A.16}$$

To check this, apply the partitioned matrix formula twice and set the first blocks equal to each other.

Standard form: $\mathbf{C} \rightarrow -\mathbf{C}^{-1}$

$$(\mathbf{A} + \mathbf{B}\mathbf{C}\mathbf{B}^\top)^{-1} = \mathbf{A}^{-1} - \mathbf{A}^{-1}\mathbf{B}(\mathbf{C}^{-1} + \mathbf{B}^\top\mathbf{A}^{-1}\mathbf{B})^{-1}\mathbf{B}^\top\mathbf{A}^{-1} \tag{A.17}$$

A.7 Lemmas on Matrix Borders

Lemma A.7.1 *Let* $\mathbf{A} \succeq 0$ *be* $n \times n$ *and* $\mathbf{x} \in \mathbb{R}^n$. *Then*

$$\hat{\mathbf{A}}_\mathbf{x} = \begin{bmatrix} \mathbf{x}^\top\mathbf{A}\mathbf{x} & \mathbf{x}^\top\mathbf{A} \\ \mathbf{A}\mathbf{x} & \mathbf{A} \end{bmatrix} \succeq 0 \tag{A.18}$$

Proof We will have proven the lemma if we can show that $\mathbf{v}^\top\hat{\mathbf{A}}_\mathbf{x}\mathbf{v} \geq 0$ for all $\mathbf{v} \in \mathbb{R}^{n+1}$. Partition $\mathbf{v}$ as $[v_0, \mathbf{v}_1]$ with $v_0 \in \mathbb{R}$ and $\mathbf{v}_1 \in \mathbb{R}^n$.

If $v_0 = 0$, then

$$\mathbf{v}^\top\hat{\mathbf{A}}_\mathbf{x}\mathbf{v} = \mathbf{v}_1^\top\mathbf{A}\mathbf{v}_1 \geq 0 \tag{A.19}$$

Assume $v_0 \neq 0$. Without loss of generality, we may assume $v_0 = 1$. Then we have

$$\begin{aligned} \mathbf{v}^\top\hat{\mathbf{A}}_\mathbf{x}\mathbf{v} = \mathbf{x}^\top\mathbf{A}\mathbf{x} + 2\mathbf{v}_1^\top\mathbf{A}\mathbf{x} + \mathbf{x}^\top\mathbf{A}\mathbf{x} &= \begin{bmatrix} \mathbf{v}_1 \\ \mathbf{x} \end{bmatrix}^\top \begin{bmatrix} \mathbf{A} & \mathbf{A} \\ \mathbf{A} & \mathbf{A} \end{bmatrix} \begin{bmatrix} \mathbf{v}_1 \\ \mathbf{x} \end{bmatrix} \\ &= \begin{bmatrix} \mathbf{v}_1 \\ \mathbf{x} \end{bmatrix}^\top \left(\begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix} \otimes \mathbf{A} \right) \begin{bmatrix} \mathbf{v}_1 \\ \mathbf{x} \end{bmatrix}. \end{aligned} \tag{A.20}$$

The last expression is greater than or equal to zero for all $\mathbf{v}_1$ because the all ones matrix and $\mathbf{A}$ are both positive semidefinite, and the tensor product of two positive semidefinite matrices is positive semidefinite. Therefore, we have shown that $\mathbf{v}^\top\hat{\mathbf{A}}_\mathbf{x}\mathbf{v} \geq 0$ as desired. ■

Lemma A.7.2 *Let* $\mathbf{A} \succeq 0$ *be* $n \times n$, $\mathbf{x} \in \mathbb{R}^n$, *and* $t \in \mathbf{R}$. *Then*

$$\begin{bmatrix} t & \mathbf{x}^\top \\ \mathbf{x} & \mathbf{A} \end{bmatrix} \succeq 0 \tag{A.21}$$

if and only if $\mathbf{x}$ *is in the range of* $\mathbf{A}$ *and* $\mathbf{x}^\top \mathbf{A}^\dagger \mathbf{x} \leq t$.

Proof Suppose $\mathbf{x}$ is not in the range of $\mathbf{A}$. Then $\mathbf{x} = \mathbf{x}_{\|} + \mathbf{x}_\perp$ with $\mathbf{x}_{\|}^\top \mathbf{x}_\perp = 0$, $\mathbf{x}_\perp \neq 0$ and $\mathbf{x}_\perp$ orthogonal to the range of $\mathbf{A}$. Since $\mathbf{A}$ is hermitian, $\mathbf{A}\mathbf{x}_\perp = 0$. Consequently, if we set

$$\mathbf{v} = -\frac{t}{\|\mathbf{x}_\perp\|^2}\mathbf{x}_\perp \tag{A.22}$$

then we have

$$\begin{aligned} \begin{bmatrix} 1 \\ \mathbf{v} \end{bmatrix}^\top \begin{bmatrix} t & \mathbf{x}^\top \\ \mathbf{x} & \mathbf{A} \end{bmatrix} \begin{bmatrix} 1 \\ \mathbf{v} \end{bmatrix} &= t + 2\mathbf{v}^\top \mathbf{x} + \mathbf{v}^\top \mathbf{A}\mathbf{v} \\ &= t - 2t + 0 = -t < 0 \end{aligned} \tag{A.23}$$

violating the assumption (A.21). Similarly, if $\mathbf{x}^\top \mathbf{A}^\dagger \mathbf{x} \geq t$, the Schur Complement Lemma tells us that (A.21) cannot hold.

The converse is an immediate consequence of A.7.1. ■

Appendix B

Equality Constrained Norm Minimization on an Arbitrary Inner Product Space

Let $\mathcal{V}$ be a real inner product space and let $\mathbf{w}_1, \ldots, \mathbf{w}_N \in \mathcal{V}$, $a_1, \ldots, a_N \in \mathbb{R}$. Consider the optimization

$$\begin{aligned} \min_{\mathbf{v} \in \mathcal{V}} \quad & \langle \mathbf{v}, \mathbf{v} \rangle \\ \text{s.t.} \quad & \langle \mathbf{v}, \mathbf{w}_i \rangle = a_i \quad \text{for } i = 1, \ldots, N \end{aligned} \tag{B.1}$$

We can construct the dual program by introducing Lagrange multipliers λ_i for $i = 1, \ldots, N$ and forming the Lagrangian

$$\mathcal{L}(\mathbf{v}, \lambda) = \langle \mathbf{v}, \mathbf{v} \rangle - 2 \sum_{i=1}^{N} \lambda_i (\langle \mathbf{v}, \mathbf{w}_i \rangle - a_i) \tag{B.2}$$

Note that, just as in the finite dimensional case, if we maximize this Lagrangian with respect to λ_i, then we either get $\langle \mathbf{v}, \mathbf{v} \rangle$ if $\langle \mathbf{v}, \mathbf{w}_i \rangle = a_i$ for all i or ∞ otherwise. Hence the problem (B.1) is equivalent to $\min_{\mathbf{v}} \max_{\lambda} \mathcal{L}(\mathbf{v}, \lambda)$. The dual program is $\max_{\lambda} \min_{\mathbf{v}} \mathcal{L}(\mathbf{v}, \lambda)$ and always achieves a lower optimal value than the primal pro-

gram. We can explicitly compute the dual as follows

$$\begin{aligned}\max_{\lambda}\min_{\mathbf{v}}\mathcal{L}(\mathbf{v},\lambda) &= \max_{\lambda}\min_{\mathbf{v}}\langle\mathbf{v},\mathbf{v}\rangle - 2\sum_{i=1}^{N}\lambda_i(\langle\mathbf{v},\mathbf{w}_i\rangle - a_i)\\ &= \max_{\lambda}\min_{\mathbf{v}}\langle\mathbf{v},\mathbf{v}\rangle - 2\langle\mathbf{v},\sum_{i=1}^{N}\lambda_i\mathbf{w}_i\rangle + 2\lambda^{\top}\mathbf{a}\\ &= \max_{\lambda}\min_{\mathbf{v}}\langle\mathbf{v}-\sum_{i=1}^{N}\lambda_i\mathbf{w}_i,\mathbf{v}-\sum_{i=1}^{N}\lambda_i\mathbf{w}_i\rangle\\ &\qquad - \langle\sum_{i=1}^{N}\lambda_i\mathbf{w}_i,\sum_{i=1}^{N}\lambda_i\mathbf{w}_i\rangle + 2\lambda^{\top}\mathbf{a}\end{aligned} \tag{B.3}$$

Since $\langle\mathbf{u},\mathbf{u}\rangle \geq 0$ for all $\mathbf{u}\in\mathcal{V}$, the minimum is achieved when

$$\mathbf{v} = \sum_{i=1}^{N}\lambda_i\mathbf{w}_i \tag{B.4}$$

This results in the dual program

$$\max_{\lambda} -\langle\sum_{i=1}^{N}\lambda_i\mathbf{w}_i,\sum_{i=1}^{N}\lambda_i\mathbf{w}_i\rangle + 2\lambda^{\top}\mathbf{a} \tag{B.5}$$

If we introduce the matrix Gram matrix of the $\mathbf{w}_i$, $\mathbf{W}$, with entries $W_{ij} = \langle\mathbf{w}_i,\mathbf{w}_j\rangle$, we can write this problem as an unconstrained convex quadratic program in N variables

$$\max_{\lambda} -\lambda^{\top}\mathbf{W}\lambda + 2\lambda^{\top}\mathbf{a} \tag{B.6}$$

The set of optimal dual solutions are the solutions of the equation $\mathbf{W}\lambda^* = \mathbf{a}$. When such a solution exists, the dual optimal value is equal to ${\lambda^*}^{\top}\mathbf{W}\lambda^*$, On the other hand, when no such solution exists, the program is unbounded above. In this case, weak duality implies that the primal program is infeasible.

When a dual optimal solution exists, consider the minimizer

$$\mathbf{v}^* = \arg\min_{\mathbf{v}}\mathcal{L}(\mathbf{v},\lambda^*) = \sum_{i=1}^{N}\lambda_i^*\mathbf{w}_i \tag{B.7}$$

First note that this vector is feasible for (B.1) because

$$\langle \mathbf{v}^*, \mathbf{w}_j \rangle = \langle \sum_{i=1}^{N} \lambda_i^* \mathbf{w}_i, \mathbf{w}_j \rangle = \sum_{i=1}^{N} \lambda_i^* W_{ij} = a_j \tag{B.8}$$

Secondly,

$$\langle \mathbf{v}^*, \mathbf{v}^* \rangle = \langle \sum_{i=1}^{N} \lambda_i^* \mathbf{w}_i \sum_{i=1}^{N} \lambda_i^* \mathbf{w}_i \rangle = \lambda^{*\top} \mathbf{W} \lambda^* \tag{B.9}$$

That is $\mathbf{v}^*$ is a primal feasible point whose primal cost is equal to the dual optimum. This implies that $\mathbf{v}^*$ is optimal for the primal and λ^* is a geometric multiplier. Furthermore, any primal optimal point $\mathbf{v}^*$ is of the form $\sum_{i=1}^{N} \lambda_i^* \mathbf{w}_i$ for some dual optimal λ^*, all of which are geometric multipliers.

Let us summarize the preceding discussion.

Theorem B.0.3 *Let $\mathcal{V}$ be a real inner product space and let $\mathbf{w}_1, \ldots, \mathbf{w}_N \in \mathcal{V}$, $\mathbf{a} \in \mathbb{R}^N$. Let $\mathbf{W}$ be the Gram matrix of the $\mathbf{w}_j$.*

(i) The equality constrained norm minimization problem

$$\begin{aligned} \min_{\mathbf{v} \in \mathcal{V}} \quad & \langle \mathbf{v}, \mathbf{v} \rangle \\ s.t. \quad & \langle \mathbf{v}, \mathbf{w}_i \rangle = a_i \quad \text{for } i = 1, \ldots, N \end{aligned} \tag{B.10}$$

has an associated dual program

$$\max_{\lambda} -\lambda^\top \mathbf{W} \lambda + 2\lambda^\top \mathbf{a} \tag{B.11}$$

which is an unconstrained convex quadratic program. The primal optimal value is equal to the dual optimal value.

(ii) Suppose the optimal value of the primal-dual pair is finite. Then the set of dual optimal solutions is given by

$$\mathcal{D} := \{\lambda \in \mathbb{R}^N \ : \ \mathbf{W}\lambda = \mathbf{a}\} \tag{B.12}$$

and the set of primal optimal solutions is given by

$$\mathcal{P} := \{\sum_{i=1}^{N} \lambda_i \mathbf{w}_i \ : \ \lambda \in \mathcal{D}\} \tag{B.13}$$

Bibliography

[1] A. Argyriou, C. A. Micchelli, and M. Pontil. Learning convex combinations of continuously parametrized basic kernels. In *Proceedings of the 18th Annual Conference on Learning Theory*, 2005.

[2] S. Arora, S. Rao, and U. Vazirani. Expander flows, geometric embeddings, and graph partitionings. In *ACM Symposium on Theory of Computing*, 2004.

[3] M. Audin. *Spinning Tops.* Number 51 in Cambridge studies in advanced mathematics. Cambridge University Press, Cambridge, UK, 1996.

[4] F. R. Bach and M. I. Jordan. Kernel independent component analysis. In *International Conference on Acoustics, Speech, and Signal Processing*, 2003.

[5] B. Bamieh, F. Paganini, and M. Dahleh. Distributed control of spatially invariant systems. *IEEE Transactions on Automatic Control*, 47(7):1091–1118, 2002.

[6] F. Barahona. On the complexity of Ising spin glass models. *Journal of Physics A*, 15:3241–3253, 1982.

[7] A. Beck and M. Teboulle. Mirror descent and nonlinear projected subgradient methods for convex optimization. *Operations Research Letters*, 31:167–175, 2003.

[8] M. Belkin and P. Niyogi. Laplacian eigenmaps and spectral techniques for embedding and clustering. In *Advances in Neural Information Processing Systems*, 2002.

[9] M. Belkin and P. Niyogi. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural Computation*, 15(6):1373–1396, 2003.

[10] A. Ben-Hur, D. Horn, H. T. Siegelmann, and V. Vapnik. Support vector clustering. *Journal of Machine Learning Research*, 2:125–137, 2001.

[11] A. Ben-Tal and A. Nemirovski. Non-euclidean restricted memory level method for large-scale convex optimization. *Mathematical Programming*, 102:407–456, 2005.

[12] D. P. Bertsekas, A. Nedic, and A. E. Ozdaglar. *Convex Analysis and Optimization.* Athena Scientific, Belmont, MA, 2003.

[13] B. E. Boser, I. M. Guyon, and V. N. Vapnik. A training algorithm for optimal margin classifiers. In *5th Annual ACM Workshop on COLT*, pages 144–152, 1992.

[14] O. Bousquet and A. Elisseeff. Stability and generalization. *Journal of Machine Learning Research*, 2(3):449–526, 2002.

[15] S. Boyd and L. Vandenberghe. *Convex Optimization.* Cambridge University Press, 2003.

[16] M. Brand. Charting a manifold. In *Neural Information Processing Systems (NIPS)*, 2002.

[17] O. Chapelle, V. Vapnik, O. Bousquet, and S. Mukherjee. Choosing multiple parameters for support vector machines. *Machine Learning*, 46(1–3):131–159, 2002.

[18] R. Cogill, S. Lall, and P. A. Parrilo. On structured semidefinite programs for the control of symmetric systems. In *Proceedings of the Allerton Conference on Communication, Control, and Computing*, 2003.

[19] F. Cucker and S. Smale. On the mathematical foundations of learning. *Bulletin of the American Mathematical Society*, 39(1):1–49, 2001.

[20] C. L. B. D. J. Newman, S. Hettich and C. J. Merz. UCI repository of machine learning databases, 1998.

[21] A. N. Dames. Position encoder. U.S. Patent No 5815091, September 1998.

[22] R. D'Andrea. A linear matrix inequality approach to decentralized control of distributed parameter systems. In *Proceedings of the 36th Conference on Decision and Control*, pages 1350–1354, 1997.

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

[24] R. D'Andrea, C. Langbort, and R. Chandra. A state space approach to control of interconnected systems. In J. Rosenthal, editor, *Mathematical Systems Theory in Biology, Communication, Computation and Finance*. Springer, IMA Book Series, 2003. To appear.

[25] T. De Bie and N. Cristianini. Convex methods for transduction. In *Neural Information Processing Systems (NIPS)*, 2003.

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

[27] D. Donoho and C. Grimes. Hessian eigenmaps: new locally linear embedding techniques for highdimensional data. Technical report, TR2003-08, Dept. of Statistics, Stanford University, 2003.

[28] G. Doretto, A. Chiuso, and Y. W. S. Soatto. Dynamic textures. *International Journal of Computer Vision (IJCV)*, 51(2):91–109, 2003.

[29] T. Evgeniou, M. Pontil, and T. Poggio. Regularization networks and support vector machines. *Advances in Computational Mathematics*, 13(1):1–50, 2000.

[30] U. Feige and G. Schectman. On the optimality of the random hyperplane rounding technique for MAX-CUT. *Random Structures and Algorithms*, 2000. to appear.

[31] A. Frieze and M. Jerrum. Improved approximation algorithms for MAX k-CUT and MAX BISECTION. In E. Balas and J. Clausen, editors, *Integer Programming and Combinatorial Optimization*, volume 920, pages 1–13. Springer, 1995.

[32] M. R. Garey and D. S. Johnson. *Computers and intractability: a guide to the theory of NP-completeness.* W. H. Freeman, San Francisco, 1978.

[33] N. Gershenfeld. Sensors for real-time cello analysis and interpretation. In *Proceedings of the ICMC*, 1991.

[34] Z. Ghahramani and S. Roweis. Learning nonlinear dynamical systems using an em algorithm. In *Neural Information Processing Systems (NIPS)*, pages 431–437, 1998.

[35] F. Girosi. An equivalence between sparse approximation and support vector machines. *Neural Computation*, 10:1455–1480, 1998.

[36] K. Glashoff and S.-Å. Gustafson. *Linear Optimization and Approximation.* Number 45 in Applied Mathematical Sciences. Springer-Verlag, New York, 1983.

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

[38] M. X. Goemans and D. P. Williamson. Approximation algorithms for MAX-3-CUT and other problems via complex semidefinite programming. In *ACM Symposium on Theory of Computing*, pages 443–452, 2001.

[39] M. Grötschel, L. Lovász, and A. Schrijver. *Geometric Algorithms and Combinatorial Optimization.* Number 2 in Algorithms and Combinatorics. Springer-Verlag, Berlin, 2nd edition, 1993.

[40] G. Gruber and F. Rendl. The bundle method for hard combinatorial optimization problems. pages 78–88, 2003.

[41] V. Guilleman and A. Pollack. *Differential Topology.* Prentice Hall, Englewood Cliffs, New Jersey, 1974.

[42] J. Ham, D. Lee, and L. Saul. Learning high dimensional correspondences from low dimensional manifolds. In *ICML*, 2003.

[43] J. Håstad. Some optimal inapproximability results. *J. ACM*, 48(4):798–859, 2001.

[44] R. Hettich and K. O. Kortanek. Semi-infinite programming: Theory, methods, and applications. *SIAM Review*, 35(3):380–429, 1993.

[45] R. A. Horn and C. R. Johnson. *Matrix Analysis.* Cambridge University Press, New York, 1985.

[46] W.-Y. Hsiang. On infinitesimal symmetrization and volume for spherical and hyperbolic tetrahedrons. *Oxford Quarterly Journal of Mathematics*, 39(2):463–368, 1988.

[47] A. Jadbabaie, N. Motee, and M. Barahona. On the stability of the kuramoto model of coupled nonlinear oscillators. In *Proceedings of the American Control Conference*, 2004.

[48] O. Jenkins and M. Mataric. A spatio-temporal extension to isomap nonlinear dimension reduction. In *International Conference on Machine Learning (ICML)*, 2004.

[49] T. Joachims. Transductive inference for text classification using support vector machines. In *International Conference on Machine Learning*, pages 200–209, 1999.

[50] D. R. Karger, R. Motwani, and M. Sudan. Approximate graph coloring by semidefinite programming. In *IEEE Symposium on Foundations of Computer Science*, pages 2–13, 1994.

[51] H. Karloff. How good is the goemans–williamson max cut algorithm? *SIAM Journal on Computing*, 29(1):336–350, 1999.

[52] H. Karloff and U. Zwick. A 7/8-approximation for MAX 3SAT. In *Proceedings of the 38th Annual IEEE Symposium on the Foundations of Computer Science*, pages 406–415, 1997.

[53] N. Karmakar. A new polynomial-time algorithm for linear programming. *Combinatorica*, 4(4):373–395, 1984.

[54] P. W. Kasteleyn. Graph theory and crystal physics. In F. Harary, editor, *Graph Theory and Theoretical Physics*, pages 43–110, London, 1967. Academic Press.

[55] J. Lafferty, A. McCallum, and F. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proc. 18th International Conf. on Machine Learning*, pages 282–289. Morgan Kaufmann, San Francisco, CA, 2001.

[56] G. R. G. Lanckriet, N. Cristianini, P. Bartlett, L. E. Ghaoui, and M. I. Jordan. Learning the kernel matrix with semidefinite programming. *Journal of Machine Learning Research*, 5:24–72, 2004.

[57] L. D. Landau and E. M. Lifshitz. *Quantum Mechanics (Non-relativistic Theory)*, volume 3 of *Course of Theoretical Physics*. Butterworth-Heinemann, Oxford, 3rd edition, 1977.

[58] J. B. Lasserre. Global optimization with polynomials and the problem of moments. *SIAM Journal on Optimization*, 11:796–817, 2001.

[59] B. Leimkuhler and S. Reich. *Simulating Hamiltonian Dynamics*. Cambridge Monographs on Applied and Computational Mathematics. Cambridge University Press, Cambridge, UK, 2004.

[60] L. Ljung. *System Identification. Theory for the user.* Prentice Hall, Upper Saddle River, NJ, 2nd edition, 1998.

[61] M. S. Lobo, L. Vandenberghe, S. Boyd, and H. Lebret. Applications of second-order cone programming. *Linear Algebra and its Applications*, 284:193–228, 1998.

[62] M. López and G. Still. Semi-infinite programming. Technical report, University of Twente, 2005.

[63] S. H. Low, J. Doyle, and F. Paganini. Internet congestion control. *IEEE Control Systems Magazine*, 2002.

[64] P.-O. Löwdin. Quantum theory of many-particle systems. i. physical interpretations by means of density matrices, natural spin-orbitals, and convergence problems in the method of configurational interaction. *Physical Review*, 97(6):1474–1489, 1954.

[65] N. C. Martins, S. Venkatesh, and M. A. Dahleh. Controller design and implementation for large-scale systems, a block decoupling approach. In *Proceedings of the American Control Conference*, pages 4728–4733, 2001.

[66] J. E. Mayer. Electron correlation. *Physical Review*, 100(6):1579–1586, 1955.

[67] D. A. Mazziotti. Variational minimization of atomic and molecular ground-state energies via the two-particle reduced density matrix. *Physical Review A*, 65(6):062511–1–062511–14, 2002.

[68] Y. Nesterov. Quality of semidefinite relaxation for nonconvex quadratic optimization. Technical report, CORE Discussion Paper 9719, 1997.

[69] P. Niyogi and F. Girosi. On the relationship between generalization error, hypothesis complexity and sample complexity for radial basis functions. *Neural Computation*, 8:819–842, 1996.

[70] P. Niyogi and F. Girosi. Generalization bounds for function approximation from scattered noisy data. *Advances in Computational Mathematics*, 10:51–80, 1999.

[71] P. V. Overschee and B. D. Moor. N4sid: Subspace algorithms for the identification of combined deterministic– stochastic systems. *Automatica*, 30:75–93, 1994.

[72] J. A. Paradiso and N. Gershenfeld. Musical applications of electric field sensing. *Computer Music Journal*, 21(2):69–89, 1997.

[73] P. Parrilo. Semidefinite programming relaxations for semialgebraic problems. *Mathematical Programming Series B*, 96(2):293–320, 2003.

[74] J. Patten, H. Ishii, J. Hines, and G. Pangaro. Sensetable: a wireless object tracking platform for tangible user interfaces. In *CHI '01: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 253–260, New York, NY, USA, 2001. ACM Press.

[75] J. Patten, B. Recht, and H. Ishii. Audiopad: A tag-based interface for musical performance. In *New Interfaces for Musical Expression*, Dublin, 2002.

[76] P. Perona and W. Freeman. A factorization approach to grouping. In *European Conference on Computer Vision (ECCV)*, volume 1406, pages 655–670, June 1998.

[77] A. Pinkus. *N-widths in approximation theory.* Springer, New York, 1985.

[78] R. Pless and I. Simon. Using thousands of images of an object. In *CVPRIP*, 2002.

[79] R. Post. Personal communication.

[80] A. Rahimi and B. Recht. Clustering with normalized cuts is clustering with a hyperplane. In *Statistical Learning in Computer Vision*, 2004.

[81] B. Recht and R. D'Andrea. Exploiting symmetry for the distributed control of spatially interconnected systems. In *42nd IEEE Conference on Decision and Control*, pages 1446–1452, 2003.

[82] B. Recht and R. D'Andrea. Distributed control of systems over discrete groups. *IEEE Transactions on Automatic Control*, 49(9):1446–1452, 2004.

[83] R. Rifkin and A. Klautau. In defense of one-vs-all classification. *Jorunal of Machine Learning Reseach*, 5:101–141, 2004.

[84] R. Rifkin, G. Yeo, and T. Poggio. *Advances in Learning Theory: Methods, Models and Applications*, volume 190 of *NATO Science Series III: Computer and Systems Sciences*, chapter Regularized Least-Squares Classification, pages 131–172. IOS Press, Amsterdam, 2003.

[85] S. Roweis and L. Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290(5500):2323–2326, 2000.

[86] I. J. Schoenberg. Metric spaces and positive definite functions. *The Annals of Mathematics*, 44(3):522–536, 1938.

[87] B. Schölkopf, A. Smola, and K.-R. Müller. Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation*, 10:1299–1319, 1998.

[88] B. Schölkopf, R. Williamson, A. Smola, J. Shawne-Taylor, and J. Platt. Support vector method for novelty detection. In *Advances in Neural Information Processing Systems 12*, 2000.

[89] J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):888–905, 2000.

[90] J. Shi and J. Malik. Normalized cuts and image segmentation. *Pattern Analysis and Machine Intelligence (PAMI)*, 22(8):888–905, 2000.

[91] M. Simonovits. How to compute the volume in high dimension? *Mathematical Programming Series B*, 97:337–374, 2003.

[92] J. R. Smith. *Electric Field Imaging.* PhD thesis, Massachusetts Institute of Technology, February 1999.

[93] A. Smola, S. Mika, B. Schoelkopf, and R. C. Williamson. Regularized principal manifolds. *Journal of Machine Learning*, 1:179–209, 2001.

[94] A. J. Smola, R. C. Williamson, S. Mika, and B. Schölkopf. Regularized principal manifolds. *Lecture Notes in Computer Science*, 1572:214–229, 1999.

[95] A. Szabo and N. S. Ostlund. *Modern Quantum Chemistry: Introduction to Advanced Electronic Structure Theory.* McGraw-Hill, New York, 1989.

[96] D. M. J. Tax and R. P. W. Duin. Support vector domain description. *Pattern Recognition Letters*, 20:1191–1199, 1999.

[97] J. B. Tenenbaum, V. de Silva, and J. C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500):2319–2323, 2000.

[98] H. Valpola and J. Karhunen. An unsupervised ensemble learning method for nonlinear dynamic state-space models. *Neural Computation*, 14(11):2647–2692, 2002.

[99] L. Vandenberghe and S. Boyd. Semidefinite programming. *SIAM Review*, 38(1):49–95, 1996.

[100] V. Vapnik. *Statistical learning theory.* Wiley, 1998.

[101] V. N. Vapnik. *The nature of statistical learning theory.* Springer, New York, 2nd edition, 2000.

[102] M. Verhaegen and P. Dewilde. Subspace model identification. *International Journal of Control*, 56(5):1187–1210, 1992.

[103] D. Verma and M. Meila. A comparison of spectral clustering algorithms. In *http://www.cs.washington.edu/research/spectral*, 2003.

[104] P. Voulgaris, G. Bianchini, and B. Bamieh. Optimal decentralized controllers for spatially invariant systems. In *Proceedings of the 39th IEEE Conference on Decision and Control*, pages 3763 –3768, 2000.

[105] G. Wahba. *Spline Models for Observational Data.* Society for Industrial and Applied Mathematics, Philadelphia, Pennsylvania, 1990.

[106] K. Weinberger and L. Saul. Unsupervised learning of image manifolds by semidefinite programming. In *Computer Vision and Pattern Recognition (CVPR)*, 2004.

[107] S. R. White. Density matrix formulation for quantum renormalization groups. *Physical Review Letters*, 69(19):2863–2866, 1992.

[108] E. Xing and M. Jordan. On semidefinite relaxation for normalized k-cut and connections to spectral clustering. Technical Report CSD-03-1265, Division of Computer Science, University of California, Berkeley, 2003.

[109] X. Zhu, Z. Ghahramani, and J. Lafferty. Semi-supervised learning using gaussian fields and harmonic functions. In *ICML*, 2003.

[110] T. Zimmerman, J. Smith, J. Paradiso, D. Allport, and N. Gershenfeld. Applying electric field sensing to human-computer interfaces. In *Proceedings of CHI'95: ACM Conference on Human Factors in Computing Systems*, 1995.