NEW DIRECTIONS IN SEMI-SUPERVISED LEARNING

by

Andrew Brian Goldberg

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

Doctor of Philosophy

(Computer Sciences)

at the

UNIVERSITY OF WISCONSIN–MADISON

2010

For my parents, who always taught me to strive for the highest achievements possible.

ABSTRACT

In many real-world learning scenarios, acquiring a large amount of labeled training data is expensive and time-consuming. Semi-supervised learning (SSL) is the machine learning paradigm concerned with utilizing unlabeled data to try to build better classifiers and regressors. Unlabeled data is a powerful resource, yet SSL can be difficult to apply in practice. The objective of this dissertation is to move the field toward more practical and robust SSL. This is accomplished by several key contributions.

First, we introduce the online (and active) semi-supervised learning setting, which considers large amounts of mostly unlabeled data arriving constantly over time. An online SSL classifier must be able to make efficient predictions at any moment and update itself in response to labeled and unlabeled data. Previously, almost all SSL assumed a fixed dataset was available before training began, and receiving new data meant retraining a potentially slow model. We present two families of online semi-supervised learners that reformulate the popular manifold and cluster assumptions into theoretically motivated and efficient online learning algorithms.

We also invent several novel model assumptions and corresponding algorithms for the more common batch SSL setting. Principled in nature, these assumptions are geared toward making SSL easier to apply to a wider variety of situations in the real world. Many SSL algorithms construct a graph over the data, to approximate an assumed (single) underlying low-dimensional manifold. In contrast, our novel multi-manifold assumption handles data lying on multiple manifolds that may differ in dimensionality, orientation, and density. The work also introduces a novel low-rank assumption, based on recent developments in matrix completion, which enables multi-label transduction with many unobserved features. Other contributions utilize several new forms of weak side information, such as dissimilarity relationships or order preferences over predictions. Finally,

SSL is applied to sentiment or opinion analysis, exploring domain-specific assumptions and graphs to extend SSL to this challenging area of natural language processing.

The dissertation provides extensive experimental results demonstrating that these novel SSL learning settings and modeling assumptions lead to algorithms with significant performance benefits in computer vision, text classification, bioinformatics, and other prediction tasks.

ACKNOWLEDGMENTS

First off, none of this would have been possible without the constant support and encouragement of my advisor Jerry. I was quite fortunate to begin my graduate career at exactly the same time that Jerry started his own next chapter as a professor here in Madison. It was his artificial intelligence course in Fall 2005 and especially the elective in advanced natural language processing and machine learning in Spring 2006 that solidified my interest in pursuing this line of research. He almost immediately took me under his wing, getting me excited about semi-supervised learning, as well as the entire research process. Jerry has always been instrumental in helping formulate ideas and master tough mathematical concepts. On numerous occasions, he has pushed me to step out of my comfort zone, and I am especially thankful for this. It was a pleasure to have been invited to be a co-author on our introductory textbook on semi-supervised learning; this experience and exposure to the larger research community is largely responsible for my upcoming job! Jerry's sustained confidence in me over the past 5 years has left a lasting mark, and I will look back on this period of my life with a great sense of accomplishment.

I must also thank my other committee members and professors who have contributed to my interest in machine learning and desire to do research. Professor Steve Wright was the first professor I ever made contact with at Wisconsin, when he personally notified me of my acceptance into the Ph.D. program. This friendly, welcoming, and unpretentious tone has permeated my entire time here. Whether in the classroom, doing research together, or figuring out how the Perl scripts on optimization-online.org work, I have appreciated getting to work closely with Steve.

Professor Jude Shavlik was also one of the first people I met upon moving to Madison. During the summer before officially enrolling, Jude encouraged me to read Tom Mitchell's classic *Machine Learning* text, which whetted my appetite for what lay ahead. Jude was also responsible for setting

me up to do an independent study with Professor Michael Ferris, in which I learned to use Matlab and code my first support vector machine. It is hard to believe how much has happened since these early experiences.

Professor Mark Craven has also been a lasting influence on my graduate career. He and Professor David Page's sequence of bioinformatics courses really got me excited about the potential applications of machine learning. I have since tried to ensure my research remains practical-minded, though without sacrificing mathematical sophistication or justification. Working with Mark in courses, TREC Genomics, and other projects has always been a pleasure. I am also especially grateful for his frequent presence and helpful feedback at various practice talks and presentations.

I am thankful for the experience of working closely with ECE Professor Rob Nowak over the last three or four years. Applying ideas from network tomography to the seemingly unrelated task of reassembling texts deconstructed into bags of words provided my first opportunity for interdepartmental collaboration and helped broaden my research perspective. I have enjoyed getting to known Rob and his students who bring a different set of skills and technical background to the table (or should I say whiteboard?). Rob has been directly involved in several of the projects represented in this thesis, and has often been like a second advisor to me, inviting me to participate in his reading groups and private group workshops.

Grace Wahba served on my preliminary exam committee and has also been an inspiration. Her course in Reproducing Kernel Hilbert Spaces forced me to push my limits, and as a result, I have come to appreciate the long history of research in Statistics that forms the foundation for most of modern machine learning.

Several mentors from outside the university have played an important role in preparing me for completing this dissertation. My summer internships with Peng Xu at Google Research, and with Ariel Fuxman and Anitha Kannan at Microsoft Research Silicon Valley, provided great hands-on exposure trying to tackle real-world problems. I enjoyed open access to vast resources, including large stores of unlabeled data and computing power. Peng, Ariel, and Anitha taught me a lot about how research gets done in the ”real world,” which has stayed with me as I finished my dissertation research and planned for the future.

In addition to those named above, I could not have made it this far without the help of many co-authors on the work presented here, as well as other projects along the way. In alphabetical order (with current affiliations in parentheses): Rakesh Agrawal (MSR), David Andrzejewski (UW), Charles R. Dyer (UW), Mohamed Eldawy (Google), Nathanael Fillmore (UW), Alex Furger (UW), Arthur Glenberg (Arizona State), Bryan Gibson (UW), Lijie Heng (UW), Tushar Khot (UW), Ming Li (Nanjing), Michael Rabbat (McGill), Ben Recht (UW), Burr Settles (CMU), John Shafer (MSR), Aarti Singh (CMU), Bradley Strock (UW), Panayiotis Tsaparas (MSR), Jurgen Van Gael (Cambridge), Junming Xu (UW), and Zhiting Xu (UW).

Throughout this process, I have benefited from the support and friendship of many other fellow classmates and colleagues in Computer Sciences and beyond. There are too many specific people to name individually, but I want to thank the members of the AI Reading Group, HAMLET (Human and Machine Learning Experiments and Theory), Graduates Anonymous, the ECE Comm-DSP reading group, and the Wednesday Night Drinking Club.

I could not have done this without the love and support of my family: my dad Steve, brother Jonathan, sister-in-law Jen, and grandparents Hilda, Evelyn, and Selig. I must also thank my aunt Martha Siegel (U. of Rochester, Ph.D. Mathematics ’69) for being an inspiration throughout graduate school. Finally, my late mother Susan would have been so proud to see her little boy get his Ph.D. She is part of what keeps me going through the challenges and frustrations of research.

Last but not least, I owe much thanks to my best friend and soon-to-be wife Amy Becker (UW-Madison, Ph.D. Mass Communications ’10). She has been a great sounding board for ideas over the last couple years, and I always enjoy our nerdy discussions about statistics and other shared research interests. Amy has helped me retain my sanity through the final stages of this endeavor, encouraging me to get things done so we can move on to the next chapter in our life together. It has been very comforting to navigate the job market and finish our dissertations together as a team; I would probably still be working on mine if it were not for her constant encouragement.

DISCARD THIS PAGE

TABLE OF CONTENTS

Page

Page

Page

DISCARD THIS PAGE

LIST OF TABLES

Table Page

DISCARD THIS PAGE

LIST OF FIGURES

PREFACE

This preface outlines the overall structure of the dissertation. I have organized the chapters into several parts described below.

Part I, Background Material

Chapter 1 begins with a quick introduction to machine learning and relevant concepts, such as regularization, that will appear throughout the dissertation. This is followed by a beginner's introduction to semi-supervised learning, providing intuitive examples to help convey the fundamental ideas behind how unlabeled data may be able to improve the learning process.

In Chapter 2, the most popular SSL modeling assumptions and corresponding families of classification algorithms are reviewed. Two of the families of algorithms discussed—graph-based methods, which make the so-called *manifold assumption*, and semi-supervised support vector machines (S3VMs), which assume the classes are separated by a low-density gap (also known as the *cluster assumption*)—play a key role in many of the remaining chapters.

Part II, Online SSL: New Learning Settings

Chapters 3 and 4 introduce two closely related novel SSL problem settings with great practical value: (1) online semi-supervised learning and (2) online *active* semi-supervised learning. Most SSL algorithms are inherently batch operations requiring all the data to be available when learning begins. My work introduces online or incremental SSL as a natural setting for many tasks. Data arrives in a streaming fashion, and most of it is unlabeled. In the active variant, the learner may

actively request the labels on some incoming data to expedite the learning process. The goal in both cases is to be able to process and learn from labeled and unlabeled items as soon as they become available; the classifier must be able to make accurate predictions at any time, while achieving bounded time and space complexity per time step to allow for life-long or infinite learning. These chapters present two radically different algorithms to achieve these goals. Both are demonstrated to work effectively on synthetic and real-world datasets.

Chapter 3 develops an online SSL framework capable of transforming any batch SSL algorithm with a convex loss function into an online SSL algorithm. This is achieved by applying online convex programming (e.g., gradient descent) to semi-supervised regularized risk minimization objective functions. After setting up this general framework, the chapter focuses on the special case of adapting manifold-based methods to work in this online setting. Naïve implementation of this idea requires storing a graph of increasing size as more data arrives. Efficient sparse approximations are introduced to remedy this problem and thus permit learning on an infinite stream of both labeled and unlabeled data.

While Chapter 3 looks at online graph or manifold-based algorithms, Chapter 4 addresses the gap or cluster assumption. Existing formulations based on this assumption involve solving a non-convex optimization problem to identify a decision boundary in a low-density region. In contrast, the fully Bayesian approach presented here introduces a new likelihood function that is sensitive to unlabeled data, and maintains an evolving estimate of the posterior distribution over the hypothesis space. Doing so allows the method to keep track of all local minima and avoid the pitfalls of existing methods that locate only a single point estimate. Bounded space complexity is achieved through particle filtering with a special Metropolis-Hastings resample-move step. Maintaining the posterior also enables the use of a principled active learning selection criterion, resulting in Online Active SemI-Supervised learning (or OASIS for short).

Part III, Batch SSL: New Model Assumptions

The five chapters in this part examine previously unexplored ways to use unlabeled data in the batch SSL setting, where a set of labeled and unlabeled data is available all at once, and the

goal is to either predict labels for the unlabeled examples (transductive learning) or leverage the unlabeled data to learn a classifier that can make more well-informed predictions on new test data (inductive learning). The aim is that the new assumptions introduced allow SSL to be applied more readily and reliably in practice, as some of the existing assumptions and formulations may be overly restrictive.

Chapter 5 relaxes the manifold assumption that is prevalent in graph-based SSL—data is typically assumed to lie on a single or multiple well-separated low-dimensional manifolds. An empirically constructed graph over items (nodes) is then used to approximate this underlying structure and define a regularizer that encourages similar predictions at nodes that are close to each other in the graph. In our work on multi-manifold SSL, we refine this assumption to allow for data lying on multiple intersecting or overlapping manifolds that may differ in dimensionality, density, and orientation. This is the case for many real-world computer vision datasets (e.g., multiple objects or people moving through space trace out arbitrary low-dimensional manifolds). The chapter introduces a novel graph-construction method that avoids placing edges between different manifolds. Backed by statistical learning theory, a simple cluster-then-label algorithm is built on top of this new Hellinger-distance-based graph. Empirical results demonstrate this new approach to graph-based SSL is both accurate and robust.

Chapter 6 introduces of a new *low-rank* assumption for SSL. We pose transductive classification as a matrix completion problem and leverage recent advances in nuclear norm minimization to develop efficient optimization techniques. By assuming that the underlying matrices of data items and labels are low rank, our formulation is able to handle three problems simultaneously: (i) multi-label learning, where each item has more than one label, (ii) transduction, where most of these labels are unspecified, and (iii) missing data, where a large number of features are missing. The low-rank assumption effectively couples multiple related learning tasks, leading to improved performance over baselines that treat the tasks separately. In addition, the matrix completion formulation provides an elegant one-step solution to data imputation and label prediction.

Many SSL assumptions rely on some notion of similarity to help connect unlabeled data to labeled data. For example, in a text categorization task, two documents containing many of the

same words may be assumed to discuss the same topic and belong to the same category. As a result, unlabeled documents may be assigned a putative category label on account of words shared with other labeled documents; several unlabeled documents may be strung together and treated as stepping stones to propagate labels even to documents with no words in common with the labeled documents. However, in many applications similarity in input features (e.g., words) does not align well with the task at hand—in opinion classification, documents may share many of the same words but describe very different opinions. This dissertation therefore develops new assumptions and semi-supervised regularizers that go beyond similarity. The next few chapters consider dissimilarity, directional ordering relations, and other task-specific relationships between items to incorporate unlabeled data.

Chapter 7 modifies existing graph-based approaches to include dissimilarity between unlabeled items to improve binary and multiclass classification. For example, in an application about predicting Internet users' political views, we find that it is relatively easy to automatically find users who likely disagree with each other. However, we still do not know which view each user holds. This chapter introduces a novel regularizer that elegantly combines readily available dissimilarity information, optional similarity information, and small amounts of labeled data to significantly boost classification accuracy.

Chapter 8 shows how to introduce SSL into kernel regression problems through a new regularizer based on known or predicted ordering relations between unlabeled items. For example, in a real-estate application, we may be able to assert that house A is expected to have a *higher* selling price than house B because it contains more bedrooms. In many applications, such weak forms of knowledge involving large amounts of unlabeled items can be easily built with simple heuristics and limited manual effort. Combined with small numbers of labeled examples (e.g., houses with known selling prices), this side information can lead to significant reductions in regression error.

Finally, Chapter 9 considers the application of SSL to sentiment categorization. In particular, we show how to transform graph-based SSL, originally designed for binary classification or regression, to the rating inference problem where the goal is to predict the number of stars (e.g., 1 to 4) assigned to a movie review by examining its text alone. A novel graph is constructed that

encodes several domain-specific assumptions and allows the resulting graph-based SSL method to outperform supervised baselines. This chapter provides a clear demonstration of how SSL can be customized for a specific real-world application.

Part IV, Conclusion

Chapter 10 summarizes the key contributions made throughout the dissertation and discusses several key remaining open questions for the future of SSL research.

Please note that much of the background material in Chapters 1–2 are drawn from a book I co-authored with Professor Xiaojin (Jerry) Zhu (Zhu and Goldberg, 2009), and Chapters 3–9 are based on published papers or manuscripts under review at the time of this publication. References within each chapter give credit to co-authors where possible.

Part I

Background Material

Chapter 1

Introduction to Semi-Supervised Learning

1.1 Review of Statistical Machine Learning

We begin by providing a brief review of basic concepts in statistical machine learning, before presenting a comprehensive overview of the subfield of semi-supervised learning. After explaining the many motivations for wanting to learn with small amounts of labeled data, we discuss some of the most common methods. We try to offer intuitive explanations and key insights into how and why these methods work, while also including enough mathematical sophistication to support the material in the remaining chapters.

Basic Terminology and Notation

An instance $\mathbf{x}$ signifies a specific object and is typically represented by a D-dimensional feature vector $\mathbf{x} = (x_1, \ldots, x_D) \in \mathbb{R}^D$. Note that boldface $\mathbf{x}$ is used to denote the whole instance, and x_d to denote the d-th feature of $\mathbf{x}$. (Note: In some chapters, we use slight variations on this notation; the differences and their meaning should be clear from context.)

A collection of instances $\{\mathbf{x}_i\}_{i=1}^n = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ is a training sample and serves as the input to the learning process. We will use x_{id} to denote the i-th instance's d-th feature. In most settings, we assume these instances are independently and identically distributed (i.i.d.) according to an underlying (but unknown to us) distribution $P(\mathbf{x})$. Formally, we write this as: $\{\mathbf{x}_i\}_{i=1}^n \overset{i.i.d.}{\sim} P(\mathbf{x})$.

As the name suggests, semi-supervised learning falls somewhere on the learning spectrum between unsupervised and supervised learning. On the one hand, unsupervised learning algorithms work on a training sample with n instances $\{\mathbf{x}_i\}_{i=1}^n$. There is no teacher providing supervision as to

how individual instances should be handled—this is the defining property of unsupervised learning. Common unsupervised learning tasks include:

- clustering: separating the n instances into groups;
- novelty detection: identifying the few instances that are very different from the rest;
- dimensionality reduction: finding a lower dimensional feature vector to represent each instance, while maintaining key characteristics of the overall training sample.

In contrast, supervised learning operates on a training sample consisting of pairs $\{(\mathbf{x}_i, y_i)\}_{i=1}^n$, where y_i is the label on $\mathbf{x}_i$ provided by nature or some teacher (hence the name *supervised* learning). Such (instance, label) pairs are called labeled data, while instances alone without labels (as in unsupervised learning) are called unlabeled data.

Let the domain of instances be $\mathcal{X}$, and the domain of labels be $\mathcal{Y}$. Let $P(\mathbf{x}, y)$ be an (unknown) joint probability distribution on instances and labels $\mathcal{X} \times \mathcal{Y}$. Given a training sample $\{(\mathbf{x}_i, y_i)\}_{i=1}^n \overset{i.i.d.}{\sim} P(\mathbf{x}, y)$, supervised learning tries to find a function $f : \mathcal{X} \mapsto \mathcal{Y}$ in some function family $\mathcal{F}$, such that $f(\mathbf{x})$ predicts the true label y on future data $\mathbf{x}$, where $(\mathbf{x}, y) \overset{i.i.d.}{\sim} P(\mathbf{x}, y)$, too.

The two most common types of supervised learning problems are classification and regression. The difference lies in the domain $\mathcal{Y}$. Classification is the supervised learning problem to find a classifier f that can predict one of a set of discrete classes $\mathcal{Y}$. Regression is the problem of learning to predict a continuous value in $\mathcal{Y}$ using a learned regression function f. Most of this work will be described in terms of classification, though the methods largely apply to both problem settings.

Given these classes of problems, we can begin to describe methods for finding and comparing different f functions. By definition, the best f is the one that minimizes the following quantity

$$f^* = \underset{f \in \mathcal{F}}{\operatorname{argmin}} \, \mathbb{E}_{(\mathbf{x},y)\sim P} \left[c(\mathbf{x}, y, f(\mathbf{x})) \right], \tag{1.1}$$

where the expectation is over random test data drawn from P. $c(\cdot)$ is a loss function that determines the cost or impact of making a prediction $f(\mathbf{x})$ that is different from the true label y. Some typical loss functions will be described later in this section.

Recall that the underlying distribution $P(\mathbf{x}, y)$ is unknown to us, so it is not possible to compute the above expectation and find f^* directly. This highlights the key task (and difficulty) in statistical machine learning: induction—generalizing predictions from a finite training sample to future unseen test data. A natural first strategy to overcome this difficulty is to measure f's performance on the training data (i.e., replace the unknown expectation by the average over the training sample). This may lead to overfitting, however; the f that minimizes training error is likely to fit itself to the statistical noise in the particular training sample instead of the true relationship between $\mathcal{X}$ and $\mathcal{Y}$. As a result, the learned f will have small training sample error, but is likely to perform less well on future test data than some other predictor $\hat{f} \in \mathcal{F}$.

Research in computational learning theory studies the issue of overfitting and establishes rigorous connections between the training sample error and the true error, using a formal notion of complexity such as the Vapnik-Chervonenkis dimension or Rademacher complexity. Informed by computational learning theory, one reasonable training strategy is to seek an f that "almost" minimizes the training sample error, while "regularizing" f so that it is not too complex in a certain sense. Regularization will play a key role in this work, as unlabeled data often enters a semi-supervised learning method through the regularizer.

Before exploring regularization in more detail, let us introduce one final basic concept: the test sample, a separate sample of labeled instances $\{(\mathbf{x}_j, y_j)\}_{j=n+1}^{n+m} \overset{i.i.d.}{\sim} P(\mathbf{x}, y)$ that can be used to estimate f's future performance. A test sample is held aside and not used during training, and therefore provides an unbiased estimate of future performance.

Regularization

Recall that, in general, we can define a loss function to specify the cost of mistakes in prediction. Formally, a loss function $c(\mathbf{x}, y, f(\mathbf{x})) \in [0, \infty)$ measures the amount of loss, or cost, of the prediction $f(\mathbf{x})$ for instance $\mathbf{x}$ with true label y. For example, in regression we can define the squared loss $c(\mathbf{x}, y, f(\mathbf{x})) = (y - f(\mathbf{x}))^2$. In classification we can define the 0/1 loss as $c(\mathbf{x}, y, f(\mathbf{x})) = 1$ if $y \neq f(\mathbf{x})$, and 0 otherwise. Alternatively, we may train a classifier based

on hinge loss: $c(\mathbf{x}, y, f(\mathbf{x})) = \max(1 - yf(\mathbf{x}), 0)$ (assuming each label $y \in \{-1, 1\}$). See Figure 2.4(a) for an illustration. In some applications (e.g., medical), the loss can depend on the specific type of misclassification or the specific instance $\mathbf{x}$.

The empirical risk of f is the average loss incurred by f on a labeled training sample: $\hat{R}(f) = \frac{1}{l}\sum_{i=1}^{l} c(\mathbf{x}_i, y_i, f(\mathbf{x}_i))$. As mentioned above, it may seem natural to find the f that minimizes the empirical risk (i.e., the principle of empirical risk minimization (ERM)):

$$f^{ERM} = \operatorname*{argmin}_{f \in \mathcal{F}} \hat{R}(f), \tag{1.2}$$

where $\mathcal{F}$ is the set of all hypotheses we consider. For classification with 0/1 loss, ERM is equivalent to minimizing the training sample error. However, f^{ERM} can overfit the particular training sample. As a consequence, f^{ERM} is not necessarily the classifier in $\mathcal{F}$ with the smallest true risk on future data.

One remedy for overfitting is to regularize the empirical risk by a regularizer $\Omega(f)$. The regularizer $\Omega(f)$ is a non-negative functional, i.e., it takes a function f as input and outputs a non-negative real value. If f is "simple" or "smooth" in some sense, $\Omega(f)$ will be close to zero; if f is too "wiggly" (i.e., it overfits and attempts to pass through all labeled training instances), $\Omega(f)$ will be large.

The regularized risk is the weighted sum of the empirical risk and the regularizer, with weight $\lambda \geq 0$: $\hat{R}(f)+\lambda\Omega(f)$. The principle of regularized risk minimization is to find the f that minimizes the regularized risk:

$$f^* = \operatorname*{argmin}_{f \in \mathcal{F}} \hat{R}(f) + \lambda\Omega(f). \tag{1.3}$$

The success of regularized risk minimization depends on the regularizer $\Omega(f)$. Different regularizers imply different assumptions of the task. For example, a commonly used regularizer for $f(\mathbf{x}) = \mathbf{w}^\top\mathbf{x}$ is $\Omega(f) = \frac{1}{2}\|\mathbf{w}\|^2$. This particular regularizer penalizes the squared norm of the parameters $\mathbf{w}$. It is helpful to view f as a point whose coordinates are determined by $\mathbf{w}$ in the parameter space. An equivalent form for the optimization problem in (1.3) is

$$\begin{aligned} \min_{f \in \mathcal{F}} \quad & \hat{R}(f) \\ \text{subject to} \quad & \Omega(f) \leq s, \end{aligned} \tag{1.4}$$

where s is determined by λ. Note the regularization term in (1.3) has been converted to an inequality constraint in (1.4). This formulation makes it easier to see that the squared norm regularizer constrains the radius of a ball in the parameter space (i.e., the constraint is simply $\frac{1}{2}\|\mathbf{w}\|^2 \leq s$). Within the ball, the function f (parameterized by $\mathbf{w}$) that best fits the training data is chosen. This controls the complexity of f and limits overfitting.

This concludes the brief overview of standard terminology and notation for statistical machine learning, including unsupervised and supervised learning. We are now ready to introduce semi-supervised learning in more detail.

1.2 Learning with Labeled and Unlabeled Data

Semi-supervised learning falls somewhere between unsupervised and supervised learning. In fact, most semi-supervised learning strategies are based on extending either unsupervised or supervised learning to include additional information typical of the other learning paradigm. Specifically, semi-supervised learning encompasses several different settings, including:

- *Semi-supervised classification and regression.* Also known as classification or regression with labeled and unlabeled data (or partially labeled data), these are extensions to the supervised problems of the same name. The training data consists of both l labeled instances $\{(\mathbf{x}_i, y_i)\}_{i=1}^{l}$ and u unlabeled instances $\{\mathbf{x}_j\}_{j=l+1}^{l+u}$. In classification, the labels are discrete, while regression operates on real-valued labels. One typically assumes that there is much more unlabeled data than labeled data, i.e., $u \gg l$. The goal of semi-supervised classification (regression) is to train a classifier (regressor) f from both the labeled and unlabeled data, such that it is better than a supervised learner trained on the labeled data alone.

- *Constrained clustering*. This is an extension to unsupervised clustering. The training data consists of unlabeled instances $\{x_i\}_{j=1}^{n}$, as well as some "supervised information" about the clusters. For example, such information can be so-called *must-link* constraints, which specify that two instances $\mathbf{x}_i, \mathbf{x}_j$ must be in the same cluster, and *cannot-link* constraints, which indicate $\mathbf{x}_i, \mathbf{x}_j$ cannot be in the same cluster. One can also constrain the size of the

clusters based on limited supervision. The goal of constrained clustering is to obtain a better clustering than the clustering from unlabeled data alone.

There are other semi-supervised learning settings, too, including dimensionality reduction (labeled instances' reduced feature representation is given), learning from positive and unlabeled data (no negative labeled instances), and so on. This work will focus on semi-supervised classification and regression. For clarity, though, this overview section will concentrate on classification. Constrained clustering is discussed in detail in the book by Basu et al. (2008) and will be described briefly in Chapter 7.

Historically, the study of semi-supervised learning has been motivated largely by its practical value in building better classifiers and regressors than supervised learning and/or at a reduced labeling cost. Recently, researchers have also begun to consider semi-supervised learning's theoretical value in understanding learning in both machines and humans. Notice the striking parallel: children learn concepts from a combination of parental feedback (labeled data) and unsupervised observations of the world around them (unlabeled data). We focus on the former motivation in this work; see Chapter 7 of Zhu and Goldberg (2009) for a review of recent work in bridging human and machine learning.

1.3 The Practical Value of Semi-Supervised Learning

Semi-supervised learning has tremendous practical value. In many tasks, there is a dearth of labeled data. The labels y may be difficult to obtain because they require human annotators, special devices, or expensive and slow experiments. For example,

- In query intent classification (Fuxman et al., 2009), an instance $\mathbf{x}$ is a Web search query (a few words of text), and the label y is an indication of the search user's intent (e.g., buy a product, find a map, etc). Since queries tend to be short and ambiguous, accurate labeling often requires examining search results and the user's clicking behavior. Large search companies are interested in millions of queries, so this can be a tedious and time-consuming process.

- In speech recognition, an instance $\mathbf{x}$ is a speech utterance, and the label y is the corresponding transcript. Accurate transcription by human expert annotators can be extremely time consuming: it took as long as 400 hours to transcribe 1 hour of speech at the phonetic level for the Switchboard telephone conversational speech data (Godfrey et al., 1992) (recordings of randomly paired participants discussing various topics such as social, economic, political, and environmental issues).

- In natural language parsing, an instance $\mathbf{x}$ is a sentence, and the label y is the corresponding parse tree. The training data, consisting of (sentence, parse tree) pairs, is known as a treebank. Treebanks are time consuming to construct, and require the expertise of linguists. For a mere 4000 sentences in the Penn Chinese Treebank, experts took two years to manually create the corresponding parse trees (Xue and Palmer, 2005).

- In spam filtering, an instance $\mathbf{x}$ is an email, and the label y is the user's judgment (spam or ham). In this situation, the bottleneck is an average user's patience to label a large number of emails.

- In video surveillance, an instance $\mathbf{x}$ is a video frame, and the label y is the identity of the object in the video. Manually labeling the objects in a large number of surveillance video frames is tedious and time consuming.

- In protein-structure prediction, an instance $\mathbf{x}$ is a DNA sequence, and the label y is the 3D folded protein structure. It can take months of expensive laboratory work by expert crystallographers to identify the 3D structure of a single protein.

While labeled data $(\mathbf{x}, y)$ is difficult to obtain in these domains, unlabeled data $\mathbf{x}$ is available in large quantity and easy to collect: text queries can be scraped from search engine logs; speech utterances can be recorded from radio broadcasts; text sentences can be crawled from the World Wide Web; emails are sitting on the mail server; surveillance cameras run 24 hours a day; and DNA protein sequences are readily available from databases. However, traditional supervised learning methods cannot use unlabeled data in training classifiers.

Semi-supervised learning is attractive because it can potentially utilize both labeled and unlabeled data to achieve better performance than supervised learning. From a different perspective, semi-supervised learning may achieve the same level of performance as supervised learning, but with fewer labeled instances. This reduces the annotation effort, which leads to reduced cost.

1.4 How is Semi-Supervised Learning Possible?

At first glance, it might seem paradoxical that one can learn anything about a predictor $f : \mathcal{X} \mapsto \mathcal{Y}$ from unlabeled data. After all, f is about the mapping from instance $\mathbf{x}$ to label y, yet unlabeled data does not provide any examples of such a mapping. The answer lies in the assumptions one makes about the link between the distribution of unlabeled data $P(\mathbf{x})$ and the target label.

Figure 1.1 shows a simple example of semi-supervised learning. Let each instance be represented by a one-dimensional feature $x \in \mathbb{R}$. There are two classes: positive and negative. Consider the following two scenarios:

1. In supervised learning, we are given only two labeled training instances $(\mathbf{x}_1, y_1) = (-1, -)$ and $(\mathbf{x}_2, y_2) = (1, +)$, shown as the red (x) and blue (o) symbols in the figure, respectively. The best estimate of the decision boundary is obviously $\mathbf{x} = 0$: all instances with $\mathbf{x} < 0$ should be classified as $y = -$, while those with $\mathbf{x} \geq 0$ as $y = +$.

2. In addition, we are also given a large number of unlabeled instances, shown as green dots in the figure. The correct class labels for these unlabeled instances are unknown. However, we observe that they form two groups. *Under the assumption* that instances in each class form a coherent group (e.g., $p(\mathbf{x}|y)$ is a Gaussian distribution, such that the instances from each class cluster around a central mean), this unlabeled data gives us more information. Specifically, it seems that the two labeled instances are not the most prototypical examples for the classes. Our *semi-supervised* estimate of the decision boundary should be between the two groups instead, at $\mathbf{x} \approx 0.4$.

If our assumption is true, then using both labeled and unlabeled data gives us a more reliable estimate of the decision boundary. Intuitively, the distribution of unlabeled data helps to identify

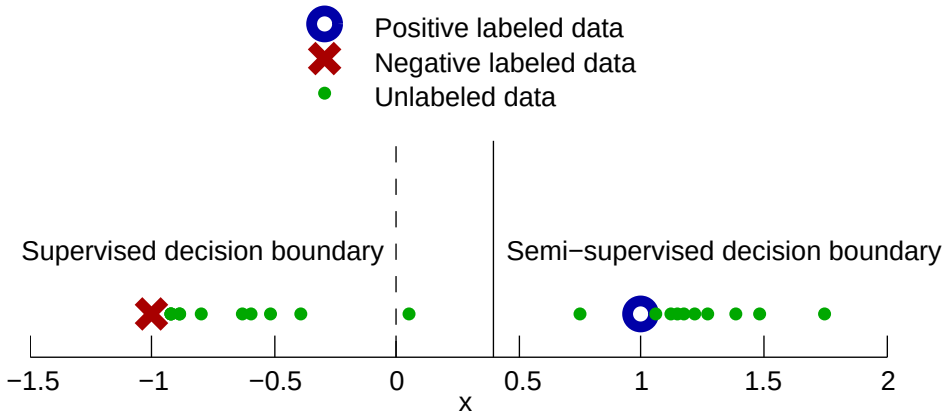


Figure 1.1: A simple example to demonstrate how semi-supervised learning is possible.

regions with the same label, and the few labeled instances then provide the actual labels. In Chapter 2, we review several other commonly used semi-supervised learning assumptions and example algorithms implementing these assumptions.

1.5 Inductive vs. Transductive Semi-Supervised Learning

There are actually two slightly different semi-supervised learning settings: inductive and transductive semi-supervised learning. In supervised classification with a fully labeled training sample, one is always interested in the performance on future test data. In semi-supervised classification, however, the training sample contains some unlabeled data, and two distinct goals are possible.

- *(Inductive)* Predict the labels on future test data: Given a training sample $\{(\mathbf{x}_i, y_i)\}_{i=1}^{l}$, $\{\mathbf{x}_j\}_{j=l+1}^{l+u}$, inductive semi-supervised learning learns a function $f : \mathcal{X} \mapsto \mathcal{Y}$ so that f is expected to be a good predictor on future data beyond $\{\mathbf{x}_j\}_{j=l+1}^{l+u}$. Like in supervised learning, one can estimate the performance on future data by using a separate test sample $\{(\mathbf{x}_k, y_k)\}_{k=1}^{m}$, which is not available during training.

- *(Transductive)* Predict the labels on the unlabeled instances in the training sample:[1] Given a training sample $\{(\mathbf{x}_i, y_i)\}_{i=1}^{l}$, $\{\mathbf{x}_j\}_{j=l+1}^{l+u}$, transductive learning trains a function $f : \mathcal{X}^{l+u} \mapsto \mathcal{Y}^{l+u}$ so that f is expected to be a good predictor on the unlabeled data $\{\mathbf{x}_j\}_{j=l+1}^{l+u}$. Note f is defined only on the given training sample, and is not required to make predictions outside. It is therefore a simpler function.

There is an interesting analogy: inductive semi-supervised learning is like an in-class exam, where the questions are not known in advance, and a student needs to prepare for all possible questions; in contrast, transductive learning is like a take-home exam, where the student knows the exam questions and needs not prepare beyond those.

1.6 Caveats

By now it should be clear why (in some cases) semi-supervised learning can use additional unlabeled data to learn a better predictor f. The key lies in the semi-supervised model *assumptions* about the link between the marginal distribution $P(\mathbf{x})$ and the conditional distribution $P(y|\mathbf{x})$. There are many different semi-supervised learning methods, and each makes slightly different assumptions about this link. These methods include self-training, probabilistic generative models, co-training, graph-based models, semi-supervised support vector machines, and so on. In the next chapter, we will introduce these methods and discuss their assumptions. Empirically, these semi-supervised learning methods do produce better classifiers than supervised learning on some datasets.

However, it is worth pointing out that blindly selecting a semi-supervised learning method for a specific task will not necessarily improve performance over supervised learning. In fact, unlabeled data can lead to *worse* performance with the wrong link assumptions. Recently, we observed evidence of this sensitivity to model assumptions in our empirical study (Goldberg and Zhu, 2009). Also, multiple researchers have informally noted that semi-supervised learning does

[1] Some authors may describe this setting equivalently as performing supervised learning with early access to the test sample (without the labels, of course). In both cases, labeled and unlabeled instances influence the learning process, and the end goal is simply to predict the labels of these specific unlabeled instances.

not *always* help. Little is written about it, though, except a few papers (Cozman et al., 2003; Elworthy, 1994). This is presumably due to "publication bias" against publishing negative results. Several examples throughout this dissertation, as well as Zhu and Goldberg (2009), test the limits of common model assumptions to highlight this sensitivity.

In short, model assumptions play a key role in semi-supervised learning. They make up for the lack of labeled data and can determine the quality of the predictor. However, making the right assumptions (or detecting wrong assumptions) remains an open question in semi-supervised learning. One aim of this work is to develop "safe" SSL methods that can mitigate the risks involved in using unlabeled data. This may be achieved through new, weaker assumptions, such as the multi-manifold learning assumption introduced in Chapter 5.

Chapter 2

Popular Semi-Supervised Learning Methods

We now summarize several popular families of semi-supervised learning methods. This is meant to highlight the variety in model assumptions, as well as set the stage for our new work described in the remaining chapters.

2.1 Self-Training

Perhaps the simplest and easiest-to-apply semi-supervised learning technique, self-training is characterized by the fact that the learning process uses its own predictions to teach itself. For this reason, it is also called self-teaching or bootstrapping (not to be confused with the statistical procedure with the same name). Self-training (Algorithm 1) can be either inductive or transductive, depending on the nature of the predictor f.

The main idea is to first train f on labeled data. The function f is then used to predict the labels for the unlabeled data. A subset S of the unlabeled data, together with their predicted labels, is then selected to augment the labeled data. Typically, S consists of the few unlabeled instances with the most confident f predictions. The function f is re-trained on the now larger set of labeled data, and the procedure repeats for some number of iterations or until some termination criterion is met. It is also possible for S to be the whole unlabeled dataset. In this case L and U remain the whole training sample, but the assigned labels on unlabeled instances might vary from iteration to iteration. One of the first successful applications of self-training was in word-sense disambiguation (Yarowsky, 1995). There have also been recent successes in natural language parsing (McClosky et al., 2006).

Input: labeled data $\{(\mathbf{x}_i, y_i)\}_{i=1}^{l}$, unlabeled data $\{\mathbf{x}_j\}_{j=l+1}^{l+u}$.

1. Initially, let $L = \{(\mathbf{x}_i, y_i)\}_{i=1}^{l}$ and $U = \{\mathbf{x}_j\}_{j=l+1}^{l+u}$.

2. Repeat:

3. Train f from L using supervised learning.

4. Apply f to the unlabeled instances in U.

5. Remove a subset S from U; add $\{(\mathbf{x}, f(\mathbf{x})) \mid \mathbf{x} \in S\}$ to L.

Algorithm 1: Generic self-training algorithm.

Self-Training Assumption: The model's own predictions, at least the high confidence ones, tend to be correct. This is likely to be the case when the classes form well-separated clusters.

The major advantages of self-training are its simplicity and the fact that it is a *wrapper* method. This means that the choice of learner for f in step 3 is left completely open. For example, the learner can be a simple kNN algorithm (see Mitchell (1997, Chapter 8)), or a very complicated classifier. The self-training procedure "wraps" around the learner without changing its inner workings. This is important for many real-world tasks like natural language processing, where the learners can be complicated black boxes not amenable to changes.

On the other hand, it is conceivable that an early mistake made by f (which is not perfect to start with, due to a small initial L) can reinforce itself by generating incorrectly labeled data. Retraining with this data will lead to an even worse f in the next iteration. Various heuristics have been proposed to alleviate this problem. Example applications of self-training, with specific details elaborated, can be found elsewhere (Maeireizo et al., 2004; Riloff et al., 2003; Rosenberg et al., 2005).

While theoretical analyses of self-training do exist for specific learning algorithms (Culp and Michailidis, 2007; Haffari and Sarkar, 2007), self-training is difficult to analyze in the more general case with an arbitrary inner classifier.

2.2 Probabilistic Generative Models

Another natural formulation of semi-supervised learning uses probabilistic generative models. Unlabeled data tells us how the instances from *all* the classes, mixed together, are distributed. If we know how the instances from *each* class are distributed, we may decompose the mixture into individual classes. This is the idea behind *mixture models* for semi-supervised learning.

Suppose we know that (or assume) the data comes from two Gaussian distributions, but we do not know their parameters (the mean, variance, and prior probabilities). We can use the data (labeled and unlabeled) to estimate these parameters for both distributions. As we saw in Figure 1.1, the labeled data can actually be misleading: the labeled instances are away from the means of the true distributions. The unlabeled data, however, helps us to identify the means of the two Gaussian distributions. Computationally, we select parameters to maximize the probability of generating such training data from the proposed model. If we only have labeled data, finding this set of parameters is straightforward, and the maximum likelihood estimate (MLE) can often be computed in closed form.

In semi-supervised learning, however, the dataset $\mathcal{D}$ consists of both labeled and unlabeled data. The likelihood depends on both the labeled and unlabeled data—this is how unlabeled data might help semi-supervised learning in mixture models. It is no longer possible to solve for the MLE analytically. Given the labeled and unlabeled data $\mathcal{D} = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_l, y_l), \mathbf{x}_{l+1}, \ldots, \mathbf{x}_{l+u}\}$, the log likelihood function (what we are trying to maximize) is defined as

$$\log p(\mathcal{D} \mid \theta) = \log \left(\prod_{i=1}^{l} p(\mathbf{x}_i, y_i \mid \theta) \prod_{i=l+1}^{l+u} p(\mathbf{x}_i \mid \theta) \right) \tag{2.1}$$

$$= \sum_{i=1}^{l} \log p(y_i \mid \theta) p(\mathbf{x}_i \mid y_i, \theta) + \sum_{i=l+1}^{l+u} \log p(\mathbf{x}_i \mid \theta), \tag{2.2}$$

where θ is a set of model parameters (e.g., Gaussian means, variances). The second term, for unlabeled instances, is the only thing that differentiates this semi-supervised log likelihood from the standard supervised log likelihood. Intuitively, a semi-supervised MLE $\hat{\theta}$ will need to fit both the labeled and unlabeled instances.

Note that the marginal probability $p(\mathbf{x} \mid \theta)$, the probability of generating $\mathbf{x}$ from any class, is defined as

$$p(\mathbf{x} \mid \theta) = \sum_{y=1}^{C} p(\mathbf{x}, y \mid \theta) = \sum_{y=1}^{C} p(y \mid \theta) p(\mathbf{x} \mid y, \theta). \tag{2.3}$$

The marginal probabilities therefore account for the fact that we know which unlabeled instances are present, but not which classes they belong to. Semi-supervised learning in mixture models amounts to optimizing or finding the MLE of (2.2).

Solving the MLE optimization problem is non-trivial. The added complexity comes from the fact that we must treat the unobserved labels $y_{l+1}, \ldots, y_{l+u}$ as hidden variables, which make the log likelihood (2.2) non-concave and hard to optimize. Fortunately, the Expectation Maximization (EM) algorithm (Dempster et al., 1977) can be used to find a (local) MLE when unlabeled data is present.[1] The EM algorithm consists of an initialization step, where model parameters are assigned initial values, and two alternating steps:

- E step: model's expected sufficient statistics are computed under the current model parameters (i.e., given some current assignment of the unlabeled instances to classes)
- M step: model parameters are updated to maximize the likelihood of observing data with these sufficient statistics (i.e., update the mean and variance of the two class distributions)

In the case of semi-supervised mixture models, the EM algorithm can be thought of as assigning "soft labels" to the unlabeled data according to the current model $\theta^{(t)}$. Because (2.2) is non-concave, EM can converge only to a local optimum, and the specific one depends on the initial parameter $\theta^{(0)}$. A common choice of $\theta^{(0)}$ is the MLE on the small labeled training set.

It is instructive to note the similarity between EM and self-training. EM can be viewed as a special form of self-training where the current classifier θ assigns both labels to the unlabeled instances, but with fractional weights $p(\mathcal{H} \mid \mathcal{D}, \theta)$, where $\mathcal{H}$ is a hidden variable indicating a class label. Unlike self-training, which usually labels only the few most confident unlabeled instances,

[1] Direct optimization methods are possible, too, for example quasi-Newton methods like L-BFGS (Liu and Nocedal, 1989).

the mixture-model approach updates the classifier based on all of the unlabeled data (fractionally assigned to both classes).

Mixture Model Assumption: The data actually comes from the mixture model, where the number of components, prior $p(y)$, and conditional $p(\mathbf{x} \mid y)$ are all correct.

Unfortunately, it can be difficult to assess the model correctness since we do not have much labeled data. Many times one would choose a generative model based on domain knowledge and/or mathematical convenience. However, if the model is wrong, semi-supervised learning could actually hurt performance (see, for example, Cozman et al., 2003). One way to alleviate this danger is to use domain knowledge to create a task-specific model (e.g., multiple components per class). Another way is to de-emphasize the unlabeled data: scale the contribution from unlabeled data in the semi-supervised log likelihood (2.2) by a small positive weight $\lambda < 1$:

$$\sum_{i=1}^{l} \log p(y_i \mid \theta) p(\mathbf{x}_i \mid y_i, \theta) + \lambda \sum_{i=l+1}^{l+u} \log p(\mathbf{x}_i \mid \theta).$$

As $\lambda \to 0$, the influence of unlabeled data vanishes and one recovers the supervised learning objective.

Mixture models provide a framework for semi-supervised learning in which the role of unlabeled data is clear. The theoretical value of labeled and unlabeled data in the context of parametric mixture models has been previously analyzed (Castelli and Cover, 1995; Ratsaby and Venkatesh, 1995). In practice, this form of semi-supervised learning can be highly effective if the generative model is (nearly) correct. In a seminal empirical paper, Nigam et al. (2000) applied a mixture of multinomial distributions for semi-supervised learning to the task of text document categorization. Similar algorithms have been successfully applied to other tasks, too (Baluja, 1998; Fujino et al., 2005, 2008). Some variations that use more than one mixture component per class, or down-weight unlabeled data relative to labeled data, can be found in the literature (Callison-Burch et al., 2004; Corduneanu and Jaakkola, 2001; Miller and Uyar, 1997; Nigam et al., 2000; Shahshahani and Landgrebe, 1994).

Input: labeled data $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_l, y_l)$, unlabeled data $\mathbf{x}_{l+1}, \ldots, \mathbf{x}_{l+u}$,
a clustering algorithm $\mathcal{A}$, and a supervised learning algorithm $\mathcal{L}$

1. Cluster $\mathbf{x}_1, \ldots, \mathbf{x}_{l+u}$ using $\mathcal{A}$.
2. For each resulting cluster, let S be the labeled instances in this cluster:
3. If S is non-empty, learn a supervised predictor from S: $f_S = \mathcal{L}(S)$.
   Apply f_S to all unlabeled instances in this cluster.
4. If S is empty, use the predictor f trained from all labeled data to label
   the unlabeled instances in this cluster.

Output: labels on unlabeled data $y_{l+1}, \ldots, y_{l+u}$.

Algorithm 2: General purpose Cluster-then-Label algorithm.

2.3 Cluster-then-Label Methods

In the preceding section, we introduced probabilistic generative models that identify mixing components from unlabeled data. Similarly, another straightforward way to perform semi-supervised learning is to first identify clusters using unlabeled data and some unsupervised clustering algorithm, then label or learn within each cluster. This high-level idea is often referred to as a Cluster-then-Label procedure for semi-supervised classification (Algorithm 2).

In step 1, the clustering algorithm $\mathcal{A}$ is unsupervised. In step 2, we learn one supervised predictor using the labeled instances that fall into each cluster, and use the predictor to label the unlabeled instances in that cluster. One can use any clustering algorithm $\mathcal{A}$ and supervised learner $\mathcal{L}$. In Chapter 5, we discuss a specific Cluster-then-Label algorithm that involves identifying clusters that correspond to different low-dimensional manifolds (Goldberg et al., 2009). Under certain conditions (El-Yaniv and Gerzon, 2005; Singh et al., 2008), theoretic analysis also justifies the Cluster-then-Label procedure (Demiriz et al., 1999; Dara et al., 2002; Goldberg et al., 2009).

Cluster-then-Label Assumption: The instances can be clustered into two or more coherent groups, such that each cluster contains only instances belonging to a single class.

2.4 Co-Training and Multiview Learning

Another broad class of semi-supervised learning methods rely on using multiple views (sets of features) or multiple classifiers. We discuss one concrete example Co-Training next, followed by a more general multiview learning framework that makes fewer assumptions.

2.4.1 Co-Training

Co-training is similar to self-training with a critical difference. In self-training, one classifier is used to make predictions on the unlabeled data, and then this data is fed back into the algorithm with predicted labels. In co-training, *two* classifiers are used, each (potentially) operating on a different view of the same instance. As an example of multiple views, consider Web page classification into `Student` or `Faculty` Web pages. In this task, the first view $\mathbf{x}^{(1)}$ can be the words on the Web page in question. The second view $\mathbf{x}^{(2)}$ can be the words in all the hyperlinks that point to the Web page. The main idea is that a classifier trained on the first view assigns predicted labels, which are given to the classifier operating on the second view, and vice versa. One can formalize this process into a *Co-Training* algorithm (Algorithm 3), similar to that which was first proposed by Blum and Mitchell (1998) and Mitchell (1999).

Note $f^{(1)}$ is a view-1 classifier: although we give it the complete feature $\mathbf{x}$, it only pays attention to the first view $\mathbf{x}^{(1)}$ and ignores the second view $\mathbf{x}^{(2)}$. $f^{(2)}$ is the other way around. They each provide their most confident unlabeled-data predictions as the training data for the other view. In this process, the unlabeled data is eventually exhausted.

Like self-training, co-training is a wrapper method that can work with any two classifiers $f^{(1)}$ and $f^{(2)}$ that can assign a confidence score to their predictions (to decide which putatively-labeled instances to add as training data). Co-training is widely applicable to many tasks, especially where it is possible to obtain two views of each instance. For examples, see Collins and Singer (1999), and Jones (2005) on named-entity classification in text processing.

Input: labeled data $\{(\mathbf{x}_i, y_i)\}_{i=1}^{l}$, unlabeled data $\{\mathbf{x}_j\}_{j=l+1}^{l+u}$, a learning speed k.

Each instance has two views $\mathbf{x}_i = [\mathbf{x}_i^{(1)}, \mathbf{x}_i^{(2)}]$.

1. Initially let the training sample be $L_1 = L_2 = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_l, y_l)\}$.
2. Repeat until unlabeled data is used up:
3. Train a view-1 classifier $f^{(1)}$ from L_1, and a view-2 classifier $f^{(2)}$ from L_2.
4. Classify the remaining unlabeled data with $f^{(1)}$ and $f^{(2)}$ separately.
5. Add $f^{(1)}$'s top k most-confident predictions $(\mathbf{x}, f^{(1)}(\mathbf{x}))$ to L_2.
   Add $f^{(2)}$'s top k most-confident predictions $(\mathbf{x}, f^{(2)}(\mathbf{x}))$ to L_1.
   Remove these from the unlabeled data.

Algorithm 3: Co-Training algorithm.

Many co-training-style algorithms exist. While the original Co-Training algorithm picks the top k most confident unlabeled instances in each view during each iteration, the so-called Co-EM algorithm (Nigam and Ghani, 2000) is less categorical and assigns fractionally-weighted class labels to all unlabeled instances (potentially different weights for each view). The step in which view 1 (2) adds *all* augmented unlabeled instances to L_2 (L_1) is equivalent to the E-step in the EM algorithm. The M-step involves updating the two views' parameters using expectations from the other view. For certain tasks, Co-EM empirically performs better than co-training. Other variations include single-view co-training (Goldman and Zhou, 2000; Chawla and Karakoulas, 2005), single-view multiple-learner Democratic Co-learning (Zhou and Goldman, 2004), Tri-Training (Zhou and Li, 2005b), and Canonical Correlation Analysis (Zhou et al., 2007).

Co-training makes several assumptions. The most obvious one is the existence of two separate views $\mathbf{x} = [\mathbf{x}^{(1)}, \mathbf{x}^{(2)}]$. For a general task, the features may not naturally split into two views. To apply co-training in this case, one can randomly split the features into two artificial views. Assuming there are two views, the success of co-training depends on the following two assumptions.

Co-Training Assumptions:

1. Each view alone is sufficient to make good classifications, given enough labeled data.

2. The two views are conditionally independent given the class label.

The first assumption is easy to understand: we need two *useful* views. The second assumption is subtle but strong: $P(\mathbf{x}^{(1)} \mid y, \mathbf{x}^{(2)}) = P(\mathbf{x}^{(1)} \mid y)$ and $P(\mathbf{x}^{(2)} \mid y, \mathbf{x}^{(1)}) = P(\mathbf{x}^{(2)} \mid y)$. In other words, if we know the true label y, then knowing one view (e.g., $\mathbf{x}^{(2)}$) does not affect what we will observe for the other view (it will simply be $P(\mathbf{x}^{(1)} \mid y)$). Thus, the instances added to one view from the other will be representative (i.e., appear to be distributed at random like a new $i.i.d.$ labeled training sample). If the assumption does not hold, the newly added instances could all be highly similar and thus be less informative for the view-1 classifier.

It can be shown that if the two assumptions hold, co-training can learn successfully from labeled and unlabeled data. When the conditional independence assumption is violated, co-training may not perform well. The assumptions have been empirically examined for some natural language processing tasks (Nigam and Ghani, 2000), and some work has investigated relaxing the conditional independence assumption (Johnson and Zhang, 2007b), since it is actually difficult to find tasks in practice in which it is satisfied.

While there is some theoretical analysis of co-training (Balcan et al., 2005b; Balcan and Blum, 2006; Dasgupta et al., 2001), it is merely a means to an end: making the two classifiers $f^{(1)}$ and $f^{(2)}$ agree (i.e., predict the same label) on the unlabeled data. Such agreement is justified by learning theory, and the intuition is simple: there are not many candidate predictors that can agree on unlabeled data in two views, so the hypothesis space is small. If a candidate predictor in this small hypothesis space also fits the labeled data well, it is less likely to be overfitting, and can be expected to be a good predictor.

2.4.2 Multiview Learning

We now discuss so-called multiview learning algorithms that explicitly enforce hypothesis agreement, without requiring explicit feature splits or the iterative mutual-teaching procedure.

These methods build on the regularized risk minimization framework introduced in Section 1.1. Recall that this approach tries to minimizes the weighted combination of an empirical training error with a regularization term. For semi-supervised learning, one can often define the regularizer $\Omega(f)$ using the unlabeled data. For example,

$$\Omega(f) = \Omega_{SL}(f) + \lambda' \Omega_{SSL}(f), \tag{2.4}$$

where $\Omega_{SL}(f)$ is a supervised regularizer, and $\Omega_{SSL}(f)$ is a semi-supervised regularizer that depends on some available unlabeled data.[2] When $\Omega_{SSL}(f)$ indeed fits the task (makes the correct assumption about what "simple" means), such regularization can produce a better f^* than that produced by $\Omega_{SL}(f)$ alone. As we will see in this section and in later ones, specific forms of Ω_{SSL} result in different semi-supervised learning algorithms.

In particular for multiview learning, $\Omega_{SSL}(f)$ can be defined to encourage agreement among multiple hypotheses. We assume the algorithm has access to k separate learners. This is the generalization of co-training to k views, hence the name multiview. However, this is a bit of a misnomer, as each learner need not use a different view. The learners might be of different types (e.g., decision tree, neural network, etc.) but take the same features of $\mathbf{x}$ as input. This is similar to an ensemble method (Opitz and Maclin, 1999). In either case, the goal is for the k learners to produce hypotheses $f_1^*, \ldots, f_k^*$ to minimize the following regularized risk:

$$\begin{aligned}(f_1^*, \ldots, f_k^*) \quad = \operatorname{argmin}_{f_1,\ldots,f_k} \quad & \sum_{v=1}^{k} \left(\sum_{i=1}^{l} c(\mathbf{x}_i, y_i, f_v(\mathbf{x}_i)) + \lambda_1 \Omega_{SL}(f_v) \right) \\ & + \lambda_2 \sum_{t=1}^{k} \sum_{v=1}^{k} \sum_{i=l+1}^{l+u} c(\mathbf{x}_i, f_t(\mathbf{x}_i), f_v(\mathbf{x}_i)).\end{aligned} \tag{2.5}$$

The intuition is for each hypothesis to not only minimize its own empirical risk, but also to agree with all the other hypotheses.

[2] Note that λ' is a weight on the SSL component in (2.4), while λ was used previously in (1.3) to refer to the weight on the entire regularizer in the regularized risk minimization objective.

The first part of the multiview regularized risk is simply the sum of individual (supervised) regularized risks. The second part defines a semi-supervised regularizer, which measures the disagreement of those k hypotheses on unlabeled instances:

$$\Omega_{SSL}(f_1, \ldots, f_k) = \sum_{t=1}^{k} \sum_{v=1}^{k} \sum_{i=l+1}^{l+u} c(\mathbf{x}_i, f_t(\mathbf{x}_i), f_v(\mathbf{x}_i)). \tag{2.6}$$

Notice pairwise disagreement is defined as the loss on an unlabeled instance x_i when pretending $f_t(x_i)$ is the label and $f_v(x_i)$ is the prediction. Such disagreement is to be minimized. The final prediction for input $\mathbf{x}$ is the label least objected to by all the hypotheses:

$$y(\mathbf{x}) = \operatorname*{argmin}_{y \in \mathcal{Y}} \sum_{v=1}^{k} c(\mathbf{x}, y, f_v^*(\mathbf{x})).$$

Different c and Ω_{SL} lead to different instantiations of multiview learning.

In a regularized risk framework, the semi-supervised learning assumption is encoded in the regularizer Ω_{SSL} (2.6) to be minimized. In this case, we assume multiple hypotheses $f_1, \ldots, f_k$ should agree with each other. However, to ensure that multiview learning is better than single-view learning, the set of agreeing hypotheses also needs to be a *small* subset of the hypothesis space $\mathcal{F}$. Consider a counter-example where all k views contain the same exact features; this will lead to k identical hypotheses that, by definition, all agree, but do not narrow down the hypothesis space. The k identical hypotheses can still be anywhere in $\mathcal{F}$. This leads to the following.

Multiview Learning Assumption: Multiview learning is effective when a set of hypotheses $f_1, \ldots, f_k$ agree with each other. Furthermore, there are not many such agreeing sets, and the agreeing set happens to have a small empirical risk.

Multiview learning was proposed as early as 1993 (de Sa, 1993). It has been applied to semi-supervised regression (Brefeld et al., 2006; Sindhwani et al., 2005b), and the more challenging problem of classification with structured outputs (Brefeld et al., 2005; Brefeld and Scheffer, 2006). Some theoretical analysis on the value of agreement among multiple learners can be found in the literature (Farquhar et al., 2006; Leskes, 2005; Sindhwani and Rosenberg, 2008; Yu et al., 2008).

2.5 Graph-Based Methods

Graph-based semi-supervised learning plays a large role in this work. Much of the work presented in the rest of this dissertation falls into this class of algorithms. In most cases, graph-based semi-supervised learning starts by constructing a graph from the training data. Given training data $\{(\mathbf{x}_i, y_i)\}_{i=1}^{l}, \{\mathbf{x}_j\}_{j=l+1}^{l+u}$, the vertices are the labeled and unlabeled instances $\{(\mathbf{x}_i)\}_{i=1}^{l} \cup \{\mathbf{x}_j\}_{j=l+1}^{l+u}$. Clearly, this is a large graph if u, the unlabeled data size, is big.

Once the graph is built, learning will involve assigning y values to the vertices in the graph. This is made possible by edges that connect labeled vertices to unlabeled vertices. The graph edges are usually undirected. An edge between two vertices $\mathbf{x}_i, \mathbf{x}_j$ traditionally represents the similarity of the two instances; some of our work extends this to include other types of relationships. Let w_{ij} be the edge weight. The idea is that if w_{ij} is large, then the two labels y_i, y_j are expected to be the same.

Before diving into specific algorithms, one can imagine a process that spreads or propagates labels from the labeled vertices to the unlabeled vertices across the graph edges, with the amount of propagation depending on the edge weights. Note that intermediate unlabeled vertices can be used as stepping stones, allowing a label to spread to unlabeled instances that are not directly connected to any labeled vertices. Therefore, the graph edge weights are of great importance. People often specify the edge weights with one of the following heuristics:

- Fully connected graph, where every pair of vertices $\mathbf{x}_i, \mathbf{x}_j$ is connected by an edge. The edge weight decreases as the Euclidean distance $\|\mathbf{x}_i - \mathbf{x}_j\|$, or some other problem-dependent distance, increases. One popular weight function is

$$w_{ij} = \exp\left(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma^2}\right), \tag{2.7}$$

  where σ is known as the bandwidth parameter and controls how quickly the weight decreases. This weight has the same form as an unnormalized Gaussian probability density function, and it is also called a Gaussian kernel or a Radial Basis Function (RBF) kernel. Note the weight is 1 when $\mathbf{x}_i = \mathbf{x}_j$, and 0 when $\|\mathbf{x}_i - \mathbf{x}_j\|$ approaches infinity.

- kNN graph. Each vertex defines its k nearest neighbor (kNN) vertices in some distance metric. Note if $\mathbf{x}_i$ is among $\mathbf{x}_j$'s kNN, the reverse is not necessarily true: $\mathbf{x}_j$ may not be among $\mathbf{x}_i$'s kNN. Typically, a symmetrized kNN graph is used: $\mathbf{x}_i$ and $\mathbf{x}_j$ are connected if one of them is among the other's kNN. This means that a vertex may have more than k edges. One may use an unweighted kNN graph: if $\mathbf{x}_i, \mathbf{x}_j$ are connected, the edge weight w_{ij} is the constant 1. The weight can also be a function of the distance as in (2.7). If $\mathbf{x}_i, \mathbf{x}_j$ are not connected, $w_{ij} = 0$. kNN graphs automatically adapt to the density of instances in feature space: in a dense region, the kNN neighborhood radius will be small; in a sparse region, the radius will be large. Empirically, kNN graphs with small k tend to perform well.

- b-matching graph. A graph constructed using the b-matching method ensures that the weights are symmetric (without the ad hoc symmetrization post-processing step used in building kNN graphs). The method solves an optimization problem to select a subset of possible edges with maximum weight (or minimum distance), subject to constraints that each vertex is connected to exactly b other vertices. Once a sparse set of edges is chosen, they may be weighted using (2.7) or set to weight 1, with all other weights equal to 0. It has recently been shown that this problem can be solved quickly using loopy belief propagation (Huang and Jebara, 2007). Also, empirical evidence (Jebara et al., 2009) suggests using balanced b-matching graphs like this can lead to significantly improved semi-supervised learning performance on some datasets.

- ϵNN graph. We connect $\mathbf{x}_i, \mathbf{x}_j$ if $\|\mathbf{x}_i - \mathbf{x}_j\| \leq \epsilon$. The edges can either be unweighted or weighted. If $\mathbf{x}_i, \mathbf{x}_j$ are not connected, $w_{ij} = 0$. ϵNN graphs are easier to construct than kNN graphs, but are less adaptive to changes in density.

These are very generic methods. Of course, better graphs can be constructed if one has knowledge of the problem domain and can define better distance functions, connectivity, and edge weights. For example, in Chapter 9, we consider edge weights based on a opinion-centric similarity function to handle the sentiment analysis task of assigning opinion ratings to movie reviews.

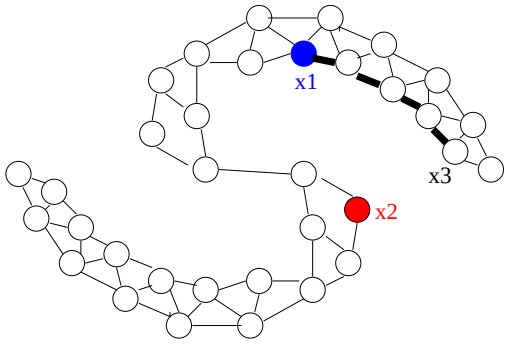


Figure 2.1: A graph constructed from labeled instances $\mathbf{x}_1, \mathbf{x}_2$ and unlabeled instances. The label of unlabeled instance $\mathbf{x}_3$ will be affected more by the label of $\mathbf{x}_1$, which is closer in the graph, than by the label of $\mathbf{x}_2$, which is farther in the graph, even though $\mathbf{x}_2$ is closer in Euclidean distance.

Figure 2.1 shows an example graph, where the edges are sparse. Let $\mathbf{x}_1, \mathbf{x}_2$ be the two labeled instances (vertices). Recall that the edges represent the "same label" assumption. For an unlabeled instance $\mathbf{x}_3$, its label y_3 is assumed to be similar to its neighbors in the graph, which in turn are similar to the neighbor's neighbors. Through this sequence of unlabeled data stepping stones, y_3 is assumed to be more similar to y_1 than to y_2. This is significant because $\mathbf{x}_3$ is in fact closer to $\mathbf{x}_2$ in Euclidean distance; without the graph, one would assume y_3 is more similar to y_2.

Formally, this intuition corresponds to estimating a label function f on the graph so that it satisfies two things: 1) the prediction $f(\mathbf{x})$ is close to the given label y on labeled vertices; 2) f should be smooth on the whole graph. This can be expressed in a regularization framework, where the former is encoded by the loss function, and the latter is encoded by a special graph-based regularizer. In the following sections, we introduce several different graph-based semi-supervised learning algorithms. They differ in the choice of the loss function and the regularizer. For simplicity, we will assume binary labels $y \in \{-1, 1\}$.

The idea of encouraging the target function to be smooth on a graph (i.e., using a graph as the basis for a regularizer) is very natural. Therefore, there are many related methods that exploit this idea, including mincut (Blum and Chawla, 2001) and randomized mincut (Blum et al., 2004), Boltzmann machines (Getz et al., 2005; Zhu and Ghahramani, 2002), graph random walk (Azran,

2007; Szummer and Jaakkola, 2001), harmonic function (Zhu et al., 2003), local and global consistency (Zhou et al., 2003), manifold regularization (Belkin et al., 2006; Sindhwani et al., 2005a, 2009), kernels from the graph Laplacian (Chapelle et al., 2002; Dai and Yeung, 2007; Kapoor et al., 2005; Kondor and Lafferty, 2002; Smola and Kondor, 2003; Zhu et al., 2004b), spectral graph transducer (Joachims, 2003), local averaging (Wang and Zhang, 2006; Wu and Schölkopf, 2007), density-based regularization (Bousquet et al., 2004; Chapelle and Zien, 2005), alternating minimization (Wang et al., 2008), boosting (Chen and Wang, 2008; Loeff et al., 2008), and the tree-based Bayes model (Kemp et al., 2003). In this section, we will discuss three of these methods: mincut, harmonic function, and manifold regularization.

Mincut Formulation

Graph-based semi-supervised learning can be formulated as a graph-cut problem (Blum and Chawla, 2001; Blum et al., 2004). Here, the positive labeled instances are treated as "source" vertices, as if some fluid is flowing out of them and through the edges. Similarly, the negative labeled instances are "sink" vertices, where the fluid would disappear. The objective is to find a minimum set of edges whose removal blocks all flow from the sources to the sinks. This defines a "cut," or a partition of the graph into two sets of vertices. The "cut size" is measured by the sum of the weights on the edges defining the cut. Once the graph is split, the vertices connecting to the sources are labeled positive, and those to the sinks are labeled negative.

Mathematically, we want to find a function $f(\mathbf{x}) \in \{-1, 1\}$ on the vertices, such that $f(\mathbf{x}_i) = y_i$ for labeled instances, and the cut size is minimized:

$$\sum_{i,j:f(\mathbf{x}_i)\neq f(\mathbf{x}_j)} w_{ij}. \tag{2.8}$$

The above quantity is the cut size: if an edge w_{ij} is removed, it must be true that $f(\mathbf{x}_i) \neq f(\mathbf{x}_j)$.

We can also cast mincut as a regularized risk minimization problem, with an appropriate loss function and regularizer:

$$\min_{f:f(\mathbf{x})\in\{-1,1\}} \infty \sum_{i=1}^{l} (y_i - f(\mathbf{x}_i))^2 + \sum_{i,j=1}^{l+u} w_{ij}(f(\mathbf{x}_i) - f(\mathbf{x}_j))^2. \tag{2.9}$$

The first term is a loss function that enforces the constraint on labeled instances: $f(\mathbf{x}_i) = y_i$. Note that we define $\infty \cdot 0 = 0$. To minimize the regularized risk, $f(\mathbf{x}_i)$ will be forced to equal y_i on labeled vertices. The second term is a regularizer corresponding to the cut size (scaled by a constant factor of 4). Recall we require $f(\mathbf{x}) \in \{-1, 1\}$ for all unlabeled vertices $\mathbf{x}$, so cut edges contribute $4w_{ij}$ to the regularizer's value. If $\mathbf{x}_i$ and $\mathbf{x}_j$ are not connected, then $w_{ij} = 0$ by definition, and if the edge exists and is not cut, then $f(\mathbf{x}_i) - f(\mathbf{x}_j) = 0$.

The mincut regularized risk problem is an integer programming problem because f is constrained to produce discrete values -1 or 1. However, efficient polynomial-time algorithms exist to solve it. It is clear that mincut is a transductive learning algorithm, because the solution f is defined only on the vertices, not on the ambient feature space (e.g., $\mathbb{R}^D$).

Harmonic Function

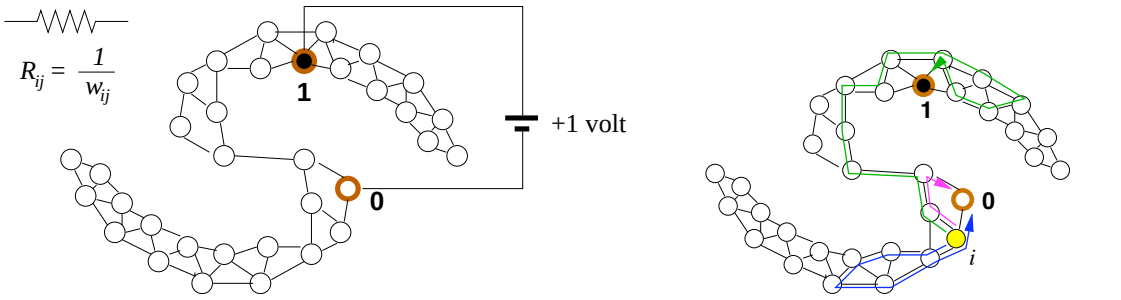
The second graph-based semi-supervised learning algorithm we introduce is the harmonic function (Zhu et al., 2003). In our context, a harmonic function is a function that has the same values as given labels on the labeled data, and satisfies the weighted average property on the unlabeled data:

$$\begin{aligned} f(\mathbf{x}_i) &= y_i, \quad i = 1 \dots l \\ f(\mathbf{x}_j) &= \frac{\sum_{k=1}^{l+u} w_{jk} f(\mathbf{x}_k)}{\sum_{k=1}^{l+u} w_{jk}}, \quad j = l+1 \dots l+u. \end{aligned} \tag{2.10}$$

In other words, the value assigned to each unlabeled vertex is the weighted average of its neighbors' values. The harmonic function is the solution to the same problem in (2.9), except that we relax f to produce real values:

$$\min_{f: f(\mathbf{x}) \in \mathbb{R}} \infty \sum_{i=1}^{l} (y_i - f(\mathbf{x}_i))^2 + \sum_{i,j=1}^{l+u} w_{ij} (f(\mathbf{x}_i) - f(\mathbf{x}_j))^2 \tag{2.11}$$

The relaxation has a profound effect: now there is a closed-form solution for f (presented at the end of this section). The solution is unique (under mild conditions) and globally optimal. The drawback of the relaxation is that in the solution, $f(\mathbf{x})$ is now a real value in $[-1, 1]$ that does not directly correspond to a label. This can however be addressed by thresholding $f(\mathbf{x})$ at zero



(a) The electric network interpretation. (b) The random walk interpretation.

Figure 2.2: The harmonic function can be interpreted as the voltages of an electric network, or the probability of reaching a positive vertex in an absorbing random walk on the graph.

to produce discrete label predictions (i.e., if $f(\mathbf{x}) >= 0$, predict $y = 1$, and if $f(\mathbf{x}) < 0$, predict $y = -1$).

The harmonic function f has many interesting interpretations. For example, one can view the graph as an electric network (see Figure 2.2(a)). Each edge is a resistor with resistance $1/w_{ij}$, or equivalently conductance w_{ij}. Suppose we connect the positive labeled vertices to the positive side of a 1-volt battery, and connect the negative vertices to the ground. Then the voltage established at each node is the harmonic function value.[3]

The harmonic function f can also be interpreted by a random walk on the graph (see Figure 2.2(b)). Imagine a particle at vertex i. In the next time step, the particle will randomly move to another vertex j with probability proportional to w_{ij}: $P(j|i) = \frac{w_{ij}}{\sum_k w_{ik}}$. The random walk continues in this fashion until the particle reaches one of the labeled vertices. This is known as an absorbing random walk, where the labeled vertices are absorbing states. Under this interpretation, the value of the harmonic function at vertex i—$f(\mathbf{x}_i)$—is the probability that a particle starting at vertex i eventually reaches a positive labeled vertex.

While a closed-form solution exists and will be presented next, there is also an iterative procedure to compute the harmonic function in (2.11). This approach may be useful for very large datasets. Initially, set $f(\mathbf{x}_i) = y_i$ for the labeled vertices $i = 1 \ldots l$, and some arbitrary value

[3]This, and the random walk interpretation below, is true when the labels $y \in \{0, 1\}$. When the labels $y \in \{-1, 1\}$, the voltages correspond to a shifted and scaled harmonic function.

for the unlabeled vertices. Iteratively update each unlabeled vertex's f value with the weighted average of its neighbors:

$$f(\mathbf{x}_i) \leftarrow \frac{\sum_{j=1}^{l+u} w_{ij} f(\mathbf{x}_j)}{\sum_{j=1}^{l+u} w_{ij}}. \tag{2.12}$$

This iterative procedure is guaranteed to converge to the harmonic function, regardless of the initial values on the unlabeled vertices. This procedure is sometimes called label propagation, as it "propagates" labels from the labeled vertices (which are fixed) gradually through the edges to all the unlabeled vertices.

Finally, let us discuss the closed-form solution for the harmonic function. The solution is easier to present using some matrix notation.

- Let W be an $(l+u) \times (l+u)$ weight matrix, whose i, j-th element is the non-negative edge weight w_{ij}. The graph is undirected, so W is a symmetric matrix.
- Let D be the $(l+u) \times (l+u)$ diagonal matrix with $D_{ii} = \sum_{j=1}^{l+u} w_{ij}$ for $i = 1 \dots l+u$. Note each diagonal entry D_{ii} is the weighted degree of vertex i (i.e., the sum of edge weights connected to i).
- The unnormalized graph Laplacian matrix L is defined as $L = D - W$, an $(l+u) \times (l+u)$ matrix.
- Let $\mathbf{f} = (f(\mathbf{x}_1), \dots, f(\mathbf{x}_{l+u}))^\top$ be the vector of f values on all vertices.

Now the regularizer in (2.11) can be written as

$$\frac{1}{2} \sum_{i,j=1}^{l+u} w_{ij} (f(\mathbf{x}_i) - f(\mathbf{x}_j))^2 = \mathbf{f}^\top L \mathbf{f}. \tag{2.13}$$

Assuming the vertices are ordered so that the labeled ones are listed first, we can partition the Laplacian matrix into four sub-matrices

$$L = \begin{bmatrix} L_{ll} & L_{lu} \\ L_{ul} & L_{uu} \end{bmatrix}. \tag{2.14}$$

We also partition $\mathbf{f}$ into $(\mathbf{f}_l, \mathbf{f}_u)$ and let $\mathbf{y}_l = (y_1, \ldots, y_l)^\top$. Then solving the constrained optimization problem using Lagrange multipliers with matrix algebra leads to the harmonic solution

$$\begin{aligned} \mathbf{f}_l &= \mathbf{y}_l \\ \mathbf{f}_u &= -{L_{uu}}^{-1} L_{ul} \mathbf{y}_l. \end{aligned} \tag{2.15}$$

Manifold Regularization

Both mincut and the harmonic function are transductive learning algorithms. They each learn a function f that is restricted to the labeled and unlabeled vertices in the graph. There is no direct way to predict the label on an unseen test instance $\mathbf{x}^*$, unless one repeats the computation after inserting a new vertex for $\mathbf{x}^*$ into the graph. This is clearly undesirable if we want predictions on a large number of test instances; we need an inductive semi-supervised learning algorithm. Another drawback of the two previous approaches is that they fix $f(\mathbf{x}) = y$ for labeled instances. It is not uncommon for real datasets to have some noisy labels, so we would like f to be able to occasionally disagree with the given labels.

Manifold regularization (Belkin et al., 2006; Sindhwani et al., 2005a) addresses these two issues. It is an inductive learning algorithm by defining f in the whole feature space: $f : \mathcal{X} \mapsto \mathbb{R}$. Formally, manifold regularization assumes that the marginal distribution $P(\mathbf{x})$ is supported on a Riemannian manifold (see Lebanon (2005, Chapter 2)). That is, even though the data is observed in a D-dimensional ambient feature space, the data really lies on a lower-dimensional manifold governed by only a few degrees of freedom. For example, handwritten digits may be represented using $16 \times 16 = 256$ pixel intensities, while intrinsically instances of the same digit class only vary according to a few underlying dimensions like rotation, size, etc. The labeled and unlabeled vertices, and hence the graph, are a random realization of the underlying manifold.

The method works as follows: f is regularized to be smooth with respect to the graph, through the use of the graph Laplacian as in (2.13). However, this regularizer alone only controls $\mathbf{f}$, the value of f on the $l + u$ training instances. To prevent f from being too wiggly (and thus having inferior generalization performance) outside the training samples, it is necessary to include a second regularization term, such as $\|f\|^2 = \int_{x \in \mathcal{X}} f(x)^2 \mathrm{d}x$. Putting them together, the regularizer for

manifold regularization becomes

$$\Omega(f) = \lambda_1 \|f\|^2 + \lambda_2 \mathbf{f}^\top L \mathbf{f}, \tag{2.16}$$

where $\lambda_1, \lambda_2 \geq 0$ control the relative strength of the two terms. To allow f to disagree with the given labels, we can simply use the squared loss function $c(\mathbf{x}, y, f(\mathbf{x})) = (y - f(\mathbf{x}))^2$, which does not greatly penalize small deviations. Other loss functions, such as hinge loss, are also possible. The complete manifold regularization problem is

$$\min_{f:\mathcal{X}\mapsto\mathbb{R}} \sum_{i=1}^{l} (y_i - f(\mathbf{x}_i))^2 + \lambda_1 \|f\|^2 + \lambda_2 \mathbf{f}^\top L \mathbf{f}. \tag{2.17}$$

The so-called representer theorem (Kimeldorf and Wahba, 1971) guarantees that the optimal f admits a finite ($l + u$, to be exact) dimensional representation. There exist efficient algorithms (e.g., quadratic programming solvers) to find the optimal f.

Beyond the unnormalized graph Laplacian matrix L, the normalized graph Laplacian matrix $\mathcal{L}$ is often used too:

$$\mathcal{L} = D^{-1/2} L D^{-1/2} = I - D^{-1/2} W D^{-1/2}. \tag{2.18}$$

This results in a slightly different regularization term

$$\mathbf{f}^\top \mathcal{L} \mathbf{f} = \frac{1}{2} \sum_{i,j=1}^{l+u} w_{ij} \left(\frac{f(\mathbf{x}_i)}{\sqrt{D_{ii}}} - \frac{f(\mathbf{x}_j)}{\sqrt{D_{jj}}} \right)^2. \tag{2.19}$$

Other variations like L^p or $\mathcal{L}^p$, where $p > 0$, are possible too. They replace the matrix L in (2.17). These all encode the same overall label-smoothness assumption on the graph, but with varying subtleties. We discuss several properties of L below.

Graph-Based Semi-Supervised Learning Assumption: The labels are "smooth" with respect to the graph, such that they vary slowly on the graph. That is, if two instances are connected by a strong edge, their labels tend to be the same.

The notion of smoothness can be made precise by spectral graph theory (Chung, 1997), which is concerned with the eigenvectors and eigenvalues of a graph, represented by its Laplacian matrix L or $\mathcal{L}$.[4] We will analyze the unnormalized Laplacian L, which has the following properties:

[4] A vector ϕ is an eigenvector of a square matrix A, if $A\phi = \lambda\phi$, where λ is the associated eigenvalue. We will focus on eigenvectors of unit length $\|\phi\| = 1$.

- L has $l+u$ eigenvalues (some may be the same) and corresponding eigenvectors $\{(\lambda_i, \phi_i)\}_{i=1}^{l+u}$. These pairs are called the graph spectrum. The eigenvectors are orthogonal: ${\phi_i}^\top \phi_j = 0$ for $i \neq j$.

- The Laplacian matrix can be decomposed into a weighted sum of outer products:

$$L = \sum_{i=1}^{l+u} \lambda_i \phi_i {\phi_i}^\top. \tag{2.20}$$

- The eigenvalues are non-negative real numbers, and can be sorted as

$$0 = \lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_{l+u}. \tag{2.21}$$

  In particular, the graph has k connected components if and only if $\lambda_1 = \ldots = \lambda_k = 0$. The corresponding eigenvectors are constant on individual connected components, and zero elsewhere.

Because the eigenvectors are orthogonal and have unit length, they form a basis in $\mathbb{R}^{l+u}$. This means any $\mathbf{f}$ on the graph can be decomposed into

$$\mathbf{f} = \sum_{i=1}^{l+u} a_i \phi_i, \tag{2.22}$$

where $a_i, i = 1 \ldots l+u$ are real-valued coefficients. After some matrix algebra, we see that the graph regularizer (2.13) can be written as

$$\mathbf{f}^\top L \mathbf{f} = \sum_{i=1}^{l+u} a_i^2 \lambda_i. \tag{2.23}$$

The regularizer is small if, for each i, either a_i or λ_i is close to zero. Intuitively, penalizing with $\mathbf{f}^\top L \mathbf{f}$ will lead to $\mathbf{f}$ solutions that assign large magnitude $|a_i|$ only to "smooth" (low frequency) basis vectors (those with small λ_i). In particular, $\mathbf{f}^\top L \mathbf{f}$ is minimized and equals zero, if $\mathbf{f}$ is in the subspace spanned by $\phi_1, \ldots, \phi_k$ for a graph with k connected components:

$$\mathbf{f} = \sum_{i=1}^{k} a_i \phi_i, \quad a_i = 0 \text{ for } i > k. \tag{2.24}$$

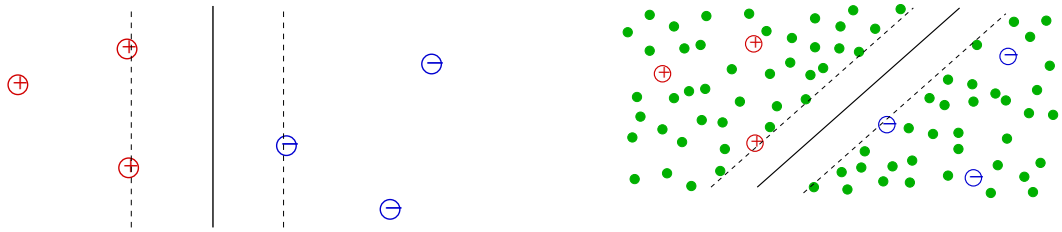
For a connected graph, only $\lambda_1 = 0$, and $\phi_1 = (1/\sqrt{l+u}, \ldots, 1/\sqrt{l+u})$. Any constant vector $\mathbf{f}$ thus has coefficients $a_1 \neq 0$, $a_i = 0$ for $i > 1$, and is a minimizer of $\mathbf{f}^\top L\mathbf{f}$. Being a constant, it is certainly the most smooth function on the graph.

Therefore, we see the connection between graph-based semi-supervised learning methods and the graph spectrum. This exposes a major weakness of this family of methods: the performance is sensitive to the graph structure and edge weights. As a result, the graph-construction problem has received considerable attention in recent years (Balcan et al., 2005a; Hein and Maier, 2006; Hein et al., 2007; Carreira-Perpinan and Zemel, 2005; Szlam et al., 2008; Zhang and Lee, 2006; Jebara et al., 2009).

Graph-based semi-supervised learning can be applied to many real-world problems, including opinion classification in text (Goldberg and Zhu, 2006; Pang and Lee, 2004)—discussed in detail in Chapter 9, word-sense disambiguation (Niu et al., 2005; Pham et al., 2005), and others (Grady and Funka-Lea, 2004; Levin et al., 2004; Krishnapuram et al., 2005). Some theoretical analyses of graph-based learning exist (Johnson and Zhang, 2007a; von Luxburg et al., 2004; Zhang and Ando, 2006).

Note that many of the graph-based semi-supervised learning algorithms have moderate to high computational complexity, often $O(u^2)$ or more. Fast computation to handle large amounts of unlabeled data is important and has been the subject of a great deal of recent research (Argyriou, 2004; Delalleau et al., 2005; Garcke and Griebel, 2005; Herbster et al., 2009; Mahdaviani et al., 2005; Sindhwani et al., 2005c; Tsang and Kwok, 2006; Yu et al., 2005; Zhu and Lafferty, 2005). Alternatively, one can perform online semi-supervised learning (Goldberg et al., 2008) where the labeled and unlabeled instances arrive sequentially. They are processed and discarded soon after to keep the computation and storage requirement low. This topic is discussed in detail in Chapters 3 and 4.

There are several extensions to the simple undirected graph that encodes similarity between vertices. In certain applications like the Web, the edges naturally are directed (Burges and Platt, 2005; Lu and Getoor, 2003; Zhou et al., 2005). Edges can also be defined on more than two vertices to form hypergraphs (Zhou et al., 2006). Some graph edges might encode dissimilarities



(a) SVM decision boundary (b) S3VM decision boundary

Figure 2.3: (a) With only labeled data, the linear decision boundary that maximizes the distance to any labeled instance is shown in solid line. Its associated margin is shown in dashed lines. (b) With additional unlabeled data, under the assumption that the classes are well-separated, the decision boundary seeks a gap in unlabeled data.

instead (Goldberg et al., 2007; Tong and Jin, 2007). The dataset can consist of multiple manifolds, requiring more advanced graph-construction methods (Goldberg et al., 2009; Wang et al., 2007; Zhou and Burges, 2007). Multiple manifolds and dissimilarity will be discussed in detail in Chapters 5 and 7, respectively.

2.6 Semi-Supervised Support Vector Machines

Finally, we discuss a natural semi-supervised learning extension to Support Vector Machines (SVMs). The intuition behind Semi-Supervised Support Vector Machines (S3VMs) is very simple. Figure 2.3(a) shows a completely labeled dataset. If asked to draw a straight line to separate the two classes, one reasonable place is right in the middle. This is the linear decision boundary found by SVMs and is shown in Figure 2.3(a). It maximizes the geometric margin—the distance to the nearest positive or negative instance—which is illustrated using dashed lines.

What if we have many additional unlabeled instances, distributed as in Figure 2.3(b)? The SVM decision boundary would cut through dense unlabeled data regions. If we assume that the two classes are well-separated, this seems undesirable. Instead, the best decision boundary now seems to be the one in Figure 2.3(b), which falls in the gap between the unlabeled data. This new decision boundary still separates the two classes in the labeled data, though its margin is smaller

than the SVM decision boundary. The new decision boundary is the one found by S3VMs, and is defined by both labeled and unlabeled data.

To formalize this intuition, we briefly review supervised SVMs and then describe S3VMs precisely. For simplicity, we will assume that there are two classes: $y \in \{-1, 1\}$. We will also assume that the decision boundary is linear in $\mathbb{R}^D$, i.e., a decision boundary is defined by the set $\{\mathbf{x}|f(\mathbf{x}) = \mathbf{w}^\top\mathbf{x} + b = 0\}$, where $\mathbf{w} \in \mathbb{R}^D$ is the parameter vector that specifies the orientation and scale of the decision boundary, and $b \in \mathbb{R}$ is an offset parameter. The decision boundary is thus defined by $f(\mathbf{x}) = 0$, and the label of $\mathbf{x}$ is predicted by $\mathrm{sign}(f(\mathbf{x}))$.

The primal SVM optimization problem can be written as an unconstrained, regularized risk minimization problem

$$\min_{\mathbf{w},b} \sum_{i=1}^{l} \max(1 - y_i(\mathbf{w}^\top\mathbf{x}_i + b), 0) + \lambda\|\mathbf{w}\|^2, \tag{2.25}$$

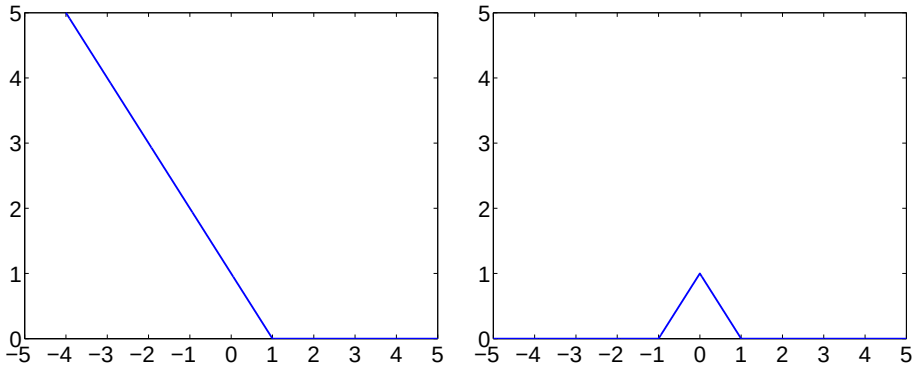
where the first term corresponds to the hinge loss function

$$c(\mathbf{x}, y, f(\mathbf{x})) = \max(1 - y(\mathbf{w}^\top\mathbf{x} + b), 0), \tag{2.26}$$

and the second term corresponds to the regularizer $\Omega(f) = \|\mathbf{w}\|^2$. The weight λ balances the two objectives. It turns out the margin can be measured as $1/\|\mathbf{w}\|$, so minimizing $\|\mathbf{w}\|^2$ is equivalent into the maximizing the margin. See Bishop (2006) for an easy-to-follow derivation of (2.25). This formulation thus attempts to find the maximum margin separation, but allows some training instances to be on the wrong side of the decision boundary.

We plot the hinge loss as a function of $yf(\mathbf{x}) = y(\mathbf{w}^\top\mathbf{x}+b)$ in Figure 2.4(a). For well-separated training instances, we have $yf(\mathbf{x}) \geq 1$. Therefore, the hinge loss penalizes instances which are on the correct side of the decision boundary, but within the margin ($0 \leq yf(\mathbf{x}) < 1$); it penalizes instances even more if they are on the wrong side of the decision boundary ($yf(\mathbf{x}) < 0$).

We can now introduce S3VMs, which were originally called Transductive Support Vector Machines (TSVMs) when proposed by Vapnik (1998), because their theory was developed to give performance bounds (theoretical guarantees) on the given unlabeled sample. However, since the learned function f naturally applies to unseen test instances, it is more appropriate to call them S3VMs.



(a) hinge loss (b) hat loss

Figure 2.4: (a) The hinge loss $c(\mathbf{x}, y, f(\mathbf{x})) = \max(1 - y(\mathbf{w}^\top \mathbf{x} + b), 0)$ as a function of $yf(\mathbf{x})$. (b) The hat loss $c(\mathbf{x}, \hat{y}, f(\mathbf{x})) = \max(1 - |\mathbf{w}^\top \mathbf{x} + b|, 0)$ as a function of $f(\mathbf{x})$.

Recall that in Figure 2.3(b), the intuition of S3VM is to place *both* labeled and unlabeled instances outside the margin. We have seen how this can be encouraged for the labeled instances using the hinge loss in Figure 2.4(a). But what about unlabeled instances? Without a label, we do not know whether an unlabeled instance $\mathbf{x}$ is on the correct or the wrong side of the decision boundary.

One way to incorporate the unlabeled instance $\mathbf{x}$ into learning is to treat the label prediction on $\mathbf{x}$, i.e., $\hat{y} = \text{sign}(f(\mathbf{x}))$, as the putative label of $\mathbf{x}$ (reminiscent of self-training). Then we can apply the hinge loss function on $\mathbf{x}$:

$$\begin{aligned} c(\mathbf{x}, \hat{y}, f(\mathbf{x})) &= \max(1 - \hat{y}(\mathbf{w}^\top \mathbf{x} + b), 0) \\ &= \max(1 - \text{sign}(\mathbf{w}^\top \mathbf{x} + b)(\mathbf{w}^\top \mathbf{x} + b), 0) \\ &= \max(1 - |\mathbf{w}^\top \mathbf{x} + b|, 0), \end{aligned} \tag{2.27}$$

where the last step follows from $\text{sign}(z)z = |z|$. The new loss function, called the hat loss due to its shape, is plotted in Figure 2.4(b). Note the x-axis is now $f(\mathbf{x})$ instead of $yf(\mathbf{x})$. Conveniently, this loss does not need the real label y; it is completely determined by $f(\mathbf{x})$.

The hat loss has a few key properties that make it desirable for semi-supervised learning. Specifically, it prefers $f(\mathbf{x}) \geq 1$ or $f(\mathbf{x}) \leq -1$ (where there is 0 loss on the "rim" of the hat).

These are instances outside the margin, far away from the decision boundary. On the other hand, it assigns a large loss value to unlabeled instances with $-1 < f(\mathbf{x}) < 1$, especially the ones with $f(\mathbf{x}) \approx 0$. These are unlabeled instances within the margin—the ones that f is uncertain about.

We now incorporate the hat loss on the unlabeled data $\{\mathbf{x}_j\}_{j=l+1}^{l+u}$ into the SVM objective (2.25) to form the S3VM objective:[5]

$$\min_{\mathbf{w},b} \sum_{i=1}^{l} \max(1 - y_i(\mathbf{w}^\top \mathbf{x}_i + b), 0) + \lambda_1 \|\mathbf{w}\|^2 + \lambda_2 \sum_{j=l+1}^{l+u} \max(1 - |\mathbf{w}^\top \mathbf{x}_j + b|, 0). \tag{2.28}$$

Clearly, the S3VM objective prefers unlabeled instances to be outside the margin. Equivalently, we want to find a decision boundary in a low density gap in the dataset, such that few unlabeled instances are close to it. Although we used the name "hat loss," it is more natural to view (2.28) as regularized risk minimization with hinge loss on labeled instances, and a regularizer involving these hat-shaped functions:

$$\Omega(f) = \lambda_1 \|\mathbf{w}\|^2 + \lambda_2 \sum_{j=l+1}^{l+u} \max(1 - |\mathbf{w}^\top \mathbf{x}_j + b|, 0). \tag{2.29}$$

Note that a class balance constraint is usually applied on top of (2.28). For poorly understood reasons, the majority (or even all) of the unlabeled instances are sometimes predicted in only one of the classes. To correct for this imbalance, one heuristic is to constrain the predicted class proportion (or sum of continuous predictions) on the unlabeled data, so that it is the same as the class proportion on the labeled data: $\frac{1}{u}\sum_{j=l+1}^{l+u} f(\mathbf{x}_j) = \frac{1}{l}\sum_{i=1}^{l} y_i$.

Finally, it is important to point out a computational difficulty of S3VMs: the objective function (2.28) is non-convex. For comparison, note that the SVM objective (2.25) is a convex function of the parameters $\mathbf{w}, b$ (i.e., due to the convexity of the hinge loss, the squared norm, and the fact that the sum of convex functions is convex). Minimizing a convex function is relatively easy, as such a function has a well-defined "bottom." On the other hand, the hat loss function is non-convex. With the sum of a large number of hat functions, the S3VM objective (2.28) is non-convex with multiple local minima. A learning algorithm can get trapped in a sub-optimal local minimum and never find the global minimum solution.

[5] Just as for SVMs, it is straightforward to form the dual problem and apply the kernel trick to S3VMs to learn non-linear classifiers.

Due to their non-convex nature, early S3VM implementations were limited by the problem size they could solve (Bennett and Demiriz, 1999; Demirez and Bennett, 2000; Fung and Mangasarian, 1999). The first widely used implementation was by Joachims (1999b), which handles the non-convexity by first assigning putative labels to the unlabeled instances and then iteratively swapping positive and negative putative labels until the objective stops decreasing. Since then, the research in S3VMs has focused on how to efficiently find a near-optimum solution (Chapelle et al., 2008). Among the many optimization techniques used are semi-definite programming (De Bie and Cristianini, 2004, 2006; Xu and Schuurmans, 2005; Xu et al., 2008), gradient search with smooth approximation to the hat function (Chapelle and Zien, 2005), deterministic annealing (Sindhwani et al., 2006), a continuation method (Chapelle et al., 2006a), the concave-convex procedure (CCCP) (Collobert et al., 2006), difference of convex (DC) programming (Wang and Shen, 2007), a fast algorithm for linear S3VMs (Sindhwani and Keerthi, 2006), Branch and Bound (Chapelle et al., 2006b), and stochastic gradient descent (combined with the manifold assumption) (Karlen et al., 2008).

S3VM Assumption: The classes are well-separated, such that the decision boundary falls into a low density region in the feature space, and does not cut through dense unlabeled data. If this assumption does not hold, this algorithm may be led astray. Some recent work relaxes the assumption on unlabeled data (Yang et al., 2009).

Several other methods also exploit the idea that unlabeled data should not be very close to the decision boundary. This intuition can be implemented in Gaussian Processes with the null category noise model (Lawrence and Jordan, 2005; Chu and Ghahramani, 2004), as information regularization (Szummer and Jaakkola, 2002; Corduneanu and Jaakkola, 2003, 2005), maximum entropy discrimination approach (Jaakkola et al., 1999), or entropy minimization (Grandvalet and Bengio, 2005; Lee et al., 2006; Mahdaviani and Choudhury, 2008). We explore another idea based on this assumption in Chapter 4.

2.7 Other Models

Many other semi-supervised learning methods and problem formulations exist in the literature, including:

- learning based on constrained clustering (Li et al., 2008);
- semi-supervised regression (Brefeld et al., 2006; Cortes and Mohri, 2006; Sindhwani et al., 2005b; Zhou and Li, 2005a);
- learning in structured output spaces, where the labels y are more complex than scalar values, e.g., sequences, graphs, etc. (Altun et al., 2005; Ando and Zhang, 2005; Brefeld and Scheffer, 2006; Lafferty et al., 2004; Taskar et al., 2003; Tsochantaridis et al., 2005; Zien et al., 2007);
- expectation regularization (Mann and McCallum, 2007), which may have deep connections with class proportion constraints (Chapelle and Zien, 2005; Chapelle et al., 2006b; Joachims, 1999b; Zhu et al., 2003);
- learning from positive and unlabeled data, when there is no negative labeled data (Denis et al., 2002; Liu et al., 2002; Lee and Liu, 2003; Elkan and Noto, 2008);
- self-taught learning (Raina et al., 2007) and the universum (Weston et al., 2006), where the unlabeled data may not come from the positive or negative classes, but rather from another third class of instances in the same general domain;
- model selection with unlabeled data (Kaariainen, 2005; Madani et al., 2005; Schuurmans and Southey, 2001), and feature selection (Li and Guan, 2008);
- inferring label sampling mechanisms (Rosset et al., 2005), multi-instance learning (Zhou and Xu, 2007), multi-task learning (Liu et al., 2008), and deep learning (Ranzato and Szummer, 2008; Weston et al., 2008);

- advances in learning theory for semi-supervised learning (Amini et al., 2009; Balcan and Blum, 2005; Cortes et al., 2008; El-Yaniv et al., 2008; Rigollet, 2007; Singh et al., 2008; Sinha and Belkin, 2008; Sokolovska et al., 2008).

Further readings on these and other semi-supervised learning topics can be found in a book collection (Chapelle et al., 2006c), a survey article (Zhu, 2005), a book written for computational linguists (Abney, 2007), and a technical report (Seeger, 2001).

Part II

Online SSL: New Learning Settings

Chapter 3

Online Manifold Regularization

We now begin describing our novel contributions to the field of semi-supervised learning. The next two chapters introduce the extremely practical setting of online semi-supervised learning, along with algorithms that operate within this regime. We consider the scenario where (mostly unlabeled) data arrives sequentially in large volume, and it is impractical to store it all before learning. When we first proposed this setting, we focused on implementing the manifold assumption in an online learning algorithm (Goldberg et al., 2008), to be discussed in the current chapter. More recently, in conjunction with the cluster or gap assumption, we consider the addition of active learning, whereby the classifier may request some specific labels. Online active semi-supervised learning (OASIS) is discussed in Chapter 4.

Consider a robot with a video camera. The robot continuously takes high frame-rate video of its surroundings, and its goal is to learn the names of various objects in the video. However, the robot receives names from humans only very rarely. The robot is thus in a semi-supervised learning situation: true labels are provided for only a small number of objects, and the rest are unlabeled.

There are several challenges that distinguish this situation from standard semi-supervised learning. The robot cannot afford to store the massive amount of mostly unlabeled video before learning; it requires an "anytime classifier" that is ready to use at all times, yet is continuously improving; training must be cheap; and since the world is changing, it should adapt to non-stationarity in classification.

These challenges are well-studied in online learning. However, our situation is also different from standard online learning. Online learning (classification) traditionally assumes that every input point is fully labeled; it cannot take advantage of unlabeled data. But in the robot case, the

vast majority of the input will be unlabeled. It seems wasteful to throw away the unlabeled input, as it may contain useful information.

We propose an online manifold regularization algorithm that differs from standard online learning in that it learns even when the input point is unlabeled. The contributions of this chapter include:

- We introduce an algorithm based on convex programming in kernel space with stochastic gradient descent, which inherits the theoretical guarantees of standard online algorithms. This combination of semi-supervised and online learning is novel. Although kernel-based online convex programming is well-understood (Zinkevich, 2003; Kivinen et al., 2004), we are not aware of prior application in the semi-supervised learning setting. To the best of our knowledge, the closest prior work is the multiview hidden Markov perceptron (Brefeld et al., 2005, Section 4), which heuristically combines multiview learning with the online perceptron. However, that work did not enjoy the theoretical guarantees afforded by the online learning literature, nor did it directly apply to other semi-supervised learning methods. In contrast, our method can lift any batch semi-supervised learning method with a convex regularized risk to the online setting. As a special case, we will discuss online manifold regularization in detail.

- Since a naïve implementation of our algorithm does not scale well, we focus on efficient, practical approximations. Specifically, we discuss two sparse approximations using buffering and online random projection trees.

- Experiments show our algorithm achieves risk and generalization accuracy comparable to standard batch manifold regularization, while each step runs quickly.

Our online semi-supervised learning setting is an interesting direction for further theoretical development, paving the way for semi-supervised learning to work on real-world life-long learning tasks.

3.1 Online Learning with Unlabeled Data

Consider an input sequence $x_1 \dots x_T$, where $x_t \in \mathbb{R}^d$ is the feature vector of the t-th data point. *Most (possibly even the vast majority) of the points are unlabeled.* Only occasionally is a point x_t accompanied by its label $y_t \in \mathcal{Y}$. This setting differs dramatically from traditional online learning where all points are labeled. Let K be a kernel over x and $\mathcal{H}_K$ the corresponding reproducing kernel Hilbert space (RKHS) (Schölkopf and Smola, 2002). Our goal is to learn a good predictor $f \in \mathcal{H}_K$ from the sequence. Importantly, learning proceeds in an iterative fashion:

1. At time t an adversary picks x_t and y_t, not necessarily from any distribution $P(x, y)$ (although we will later assume i.i.d. for predicting future data). The adversary presents x_t to the learner.

2. The learner makes prediction $f_t(x_t)$ using its current predictor f_t.

3. With a small probability p_l, the adversary reveals the label y_t. Otherwise, the adversary abstains, and x_t remains unlabeled.

4. The learner updates its predictor to f_{t+1} based on x_t and the adversary's feedback y_t, if any.

We hope the functions $f_1 \dots f_T$ "do well" on the sequence, and on future input if the data is indeed i.i.d. The exact performance criteria is defined below.

3.2 From Batch to Online Semi-Supervised Learning

Before introducing our online learning algorithm, we first review *batch* semi-supervised learning, where the learner has access to the labeled and unlabeled data all at once. Recall that a unifying framework for batch semi-supervised learning is risk minimization with specialized semi-supervised regularizers. That is, one seeks the solution $f^* = \operatorname{argmin}_{f \in \mathcal{H}_K} J(f)$, where the *batch semi-supervised regularized risk* is

$$J(f) = \frac{1}{l} \sum_{t=1}^{T} \delta(y_t) c(f(x_t), y_t) + \frac{\lambda_1}{2} \|f\|_K^2 + \lambda_2 \Omega_{SSL}(f),$$

where l is the number of labeled points, $\delta(y_t)$ is an indicator function equal to 1 if y_t is present (labeled) and 0 otherwise, c is a convex loss function, λ_1, λ_2 are regularizer weights, $\|f\|_K$ is the RKHS norm of f, and Ω_{SSL} is the semi-supervised regularizer which depends on f and $x_1 \ldots x_T$. Specific choices of Ω_{SSL} lead to familiar semi-supervised learning methods, such as manifold regularization (Belkin et al., 2006; Sindhwani et al., 2005a; Zhu et al., 2003). For this work, we consider manifold regularization's Ω_{SSL} to be

$$\Omega_{SSL} = \frac{1}{2T} \sum_{s,t=1}^{T} w_{st}(f(x_s) - f(x_t))^2.$$

Recall that w_{st} is a graph edge weight encoding the similarity between instances x_s and x_t.

A key observation is that for certain semi-supervised learning methods, the batch risk $J(f)$ is the sum of *convex functions* in f. These methods include manifold regularization and multi-view learning, but not S3VMs whose hat loss is non-convex (see Chapter 4 for an online learning method based on the same low-density gap or cluster assumption that S3VMs encodes using the hat loss). For convex semi-supervised learning methods, one can derive a corresponding online semi-supervised learning algorithm using online convex programming. The remainder of this chapter focuses on manifold regularization, with the understanding that online versions of multiview learning and other convex semi-supervised learning methods can be derived similarly.

We follow a general approach to online convex programming (Zinkevich, 2003; Kivinen et al., 2004). The batch risk for our version of manifold regularization is

$$J(f) = \frac{1}{l} \sum_{t=1}^{T} \delta(y_t) c(f(x_t), y_t) + \frac{\lambda_1}{2} \|f\|_K^2 + \frac{\lambda_2}{2T} \sum_{s,t=1}^{T} w_{st}(f(x_s) - f(x_t))^2, \tag{3.1}$$

and f^* is the batch solution that minimizes $J(f)$. In online learning, the learner only has access to the input sequence up to the current time. We thus define the *instantaneous regularized risk* $J_t(f)$ at time t to be

$$J_t(f) = \frac{T}{l} \delta(y_t) c(f(x_t), y_t) + \frac{\lambda_1}{2} \|f\|_K^2 + \lambda_2 \sum_{i=1}^{t-1} w_{it}(f(x_i) - f(x_t))^2. \tag{3.2}$$

The last term in $J_t(f)$ involves the graph edges from x_t to all previous points up to time t. The astute reader might notice that this poses a computational challenge—we will return to this issue

in Section 3.3. While T appears in (3.2), $J_t(f)$ depends only on the *ratio* T/l. This is the empirical estimate of the inverse label probability $1/p_l$, which we assume is given and easily determined based on the rate at which humans can label the data at hand.

All the J_t's are convex. They are intimately connected to the batch risk J:

Proposition 1 $J(f) = \frac{1}{T}\sum_{t=1}^{T} J_t(f)$.

Our online algorithm constructs a sequence of functions $f_1 \dots f_T$. Let $f_1 = 0$. The online algorithm simply performs a gradient descent step that aims to reduce the instantaneous risk in each iteration:

$$f_{t+1} = f_t - \eta_t \left.\frac{\partial J_t(f)}{\partial f}\right|_{f_t}. \tag{3.3}$$

The step size η_t needs to decay at a certain rate, e.g., $\eta_t = 1/\sqrt{t}$. Under mild conditions, this seemingly naïve online algorithm has a remarkable guarantee that on any input sequence, there is asymptotically "no regret" compared to the batch solution f^*. Specifically, let the *average instantaneous risk* incurred by the online algorithm be $J_{air}(T) \equiv \frac{1}{T}\sum_{t=1}^{T} J_t(f_t)$. Note J_{air} involves a varying sequence of functions $f_1 \dots f_T$. As a standard quality measure in online learning, we compare J_{air} to the risk of the best *fixed* function in hindsight:

$$J_{air}(T) - \min_f \frac{1}{T}\sum_{t=1}^{T} J_t(f) = J_{air}(T) - \min_f J(f) = J_{air}(T) - J(f^*),$$

where we used Proposition 1. This difference is known as the average regret. Applying Theorem 1 of Zinkevich (2003) results in the no-regret guarantee $\limsup_{T\to\infty} J_{air}(T) - J(f^*) \le 0$. It is in this sense that the online algorithm performs as well as the batch algorithm on the sequence.

To compute (3.3) for manifold regularization, we first express the functions $f_1 \dots f_T$ using a common set of representers $x_1 \dots x_T$ (Kimeldorf and Wahba, 1971). At any time t, however, only $t-1$ may have non-zero coefficients:

$$f_t = \sum_{i=1}^{t-1} \alpha_i^{(t)} K(x_i, \cdot). \tag{3.4}$$

The problem of finding f_{t+1} becomes computing the coefficients $\alpha_1^{(t+1)}, \dots, \alpha_t^{(t+1)}$. Again, this will be a computational issue when T is large, and will be addressed in Section 3.3. We extend the

Parameters: edge weight function w, kernel K, weights λ_1, λ_2, loss function c, label ratio T/l, step sizes η_t

Initialize $t = 1, f_1 = 0$

loop

receive x_t, predict $f_t(x_t)$ using (3.4)

(occasionally) receive y_t

update f_t to f_{t+1} using (3.6)

store x_t, let $t = t + 1$

end loop

Algorithm 4: Online Manifold Regularization

kernel online supervised learning approach of Kivinen et al. (2004) to semi-supervised learning by writing the gradient $\partial J_t(f)/\partial f$ in (3.3) as

$$\frac{T}{l}\delta(y_t)c'(f(x_t), y_t)K(x_t, \cdot) + \lambda_1 f + 2\lambda_2 \sum_{i=1}^{t-1} w_{it}(f(x_i) - f(x_t))(K(x_i, \cdot) - K(x_t, \cdot)), \quad (3.5)$$

where we used the reproducing property of RKHS in computing the derivative: $\partial f(x)/\partial f = \partial\langle f, K(x, \cdot)\rangle/\partial f = K(x, \cdot)$. c' is the (sub)gradient of the loss function c. For example, when $c(f(x), y)$ is the hinge loss $\max(1 - f(x)y, 0)$, we may define $c'(f(x), y) = -y$ if $f(x)y \leq 1$, and 0 otherwise. Putting (3.5) back in (3.3), and replacing f_t with its kernel expansion (3.4), it can be shown that f_{t+1} has the following coefficients:

$$\begin{aligned} \alpha_i^{(t+1)} &= (1 - \eta_t\lambda_1)\alpha_i^{(t)} - 2\eta_t\lambda_2 w_{it}(f_t(x_i) - f_t(x_t)), \quad i = 1 \ldots t-1 \\ \alpha_t^{(t+1)} &= 2\eta_t\lambda_2 \sum_{i=1}^{t-1} w_{it}(f_t(x_i) - f_t(x_t)) - \eta_t \frac{T}{l}\delta(y_t)c'(f(x_t), y_t). \end{aligned} \quad (3.6)$$

We now have a basic online manifold regularization algorithm; see Algorithm 4.

When the data is i.i.d., the generalization risk of the average function $\bar{f} = 1/T \sum_{t=1}^{T} f_t$ approaches that of f^* (Cesa-Bianchi et al., 2004). The average function $\bar{f}$ involves all representers $x_1, \ldots, x_T$. For basic online manifold regularization, it is possible to incrementally maintain the exact $\bar{f}$ as time increases. However, for the sparse approximations introduced below, the basis

changes over time (i.e., it is no longer simply $K(x_1, \cdot), \ldots, K(x_T, \cdot)$). Therefore, in those cases $\bar{f}$ can be maintained (approximately) using matching pursuit (Vincent and Bengio, 2002). In our experiments, we compare the classification accuracy of $\bar{f}$ vs. f^* on a separate test set, which is of practical interest.

3.3 Sparse Approximations

Unfortunately, Algorithm 4 will not work in practice because it needs to store every input point and soon runs out of memory; it also has time complexity $O(T^2)$. In particular, the instantaneous risk (3.2) and the kernel representation (3.4) both involve the sequence up to the current time. To be useful, it is imperative to sparsify both terms. In this section, we present two distinct approaches for this purpose: i) using a small buffer of points, and ii) constructing a random projection tree that represents the manifold structure and can be used to cluster the data.

3.3.1 Buffering

A buffering strategy is often used in online learning as a way to restrict the total number of points stored over time (Dekel et al., 2005). Let the buffer size be τ. The simplest buffering strategy replaces the oldest point $x_{t-\tau}$ in the buffer with the incoming point x_t. With buffering, the approximate instantaneous risk is

$$J_t(f) \;=\; \frac{T}{l}\delta(y_t)c(f(x_t), y_t) + \frac{\lambda_1}{2}\|f\|_K^2 + \lambda_2 \frac{t}{\tau}\sum_{i=t-\tau}^{t-1}(f(x_i) - f(x_t))^2 w_{it}, \tag{3.7}$$

where the scaling factor t/τ keeps the magnitude of the graph regularizer comparable to the unbuffered version. In terms of manifold regularization, buffering corresponds to a dynamic graph on the points in the buffer. Similarly, the kernel expansion now has τ terms:

$$f_t = \sum_{i=t-\tau}^{t-1} \alpha_i^{(t)} K(x_i, \cdot).$$

With buffering, the function update involves two steps. In the first step, we update f_t to an intermediate function f' represented by a basis of $\tau + 1$ elements, consisting of the old buffer and the

new point x_t:

$$\begin{aligned} f' &= \sum_{i=t-\tau}^{t-1} \alpha'_i K(x_i, \cdot) + \alpha'_t K(x_t, \cdot) \\ \alpha'_i &= (1 - \eta_t \lambda_1)\alpha_i^{(t)} - 2\eta_t \lambda_2 (f_t(x_i) - f_t(x_t)) w_{it}, \;\; i = t-\tau \ldots t-1 \\ \alpha'_t &= 2\eta_t \lambda_2 \frac{t}{\tau} \sum_{i=t-\tau}^{t-1} (f_t(x_i) - f_t(x_t)) w_{it} - \eta_t \frac{T}{l} \delta(y_t) c'(f(x_t), y_t). \end{aligned} \tag{3.8}$$

Second, we evict $x_{t-\tau}$ from the buffer, add x_t to the buffer, and approximate f' (which uses $\tau + 1$ basis functions) with f_{t+1} (which uses τ basis functions):

$$\min_{\alpha^{(t+1)}} \|f' - f_{t+1}\|^2 \quad \text{s.t. } f_{t+1} = \sum_{i=t-\tau+1}^{t} \alpha_i^{(t+1)} K(x_i, \cdot). \tag{3.9}$$

Intuitively, we "spread" $\alpha'_{t-\tau} K(x_{t-\tau}, \cdot)$ to the remaining points in the buffer, in an attempt to minimize the change caused by truncation. We use kernel matching pursuit (Vincent and Bengio, 2002) to efficiently find the approximate coefficients $\alpha^{(t+1)}$ in (3.9). Matching pursuit is a greedy function approximation scheme. It iteratively selects a basis function on which to spread the residual in $\alpha'_{t-\tau} K(x_{t-\tau}, \cdot)$. The number of steps (i.e., basis functions selected) can be controlled to trade-off approximation error and speed. We run matching pursuit until the norm of the residue vector has been sufficiently reduced. We call the above buffering strategy "**buffer**." The overall time complexity for buffering is $O(T)$.

An alternative buffering strategy, "**buffer-U**," evicts the oldest *unlabeled* points in the buffer while keeping as many labeled points as possible. This is motivated by the fact that the labeled points tend to have larger α coefficients and exert more influence on our learned function. The oldest labeled point is evicted from the buffer only when it is filled with labeled points. Note this is distinct from batch learning: the labeled points only form a better basis, but learning is still done via gradient descent.

3.3.2 Random Projection Tree

Another way to improve Algorithm 4 is to construct a sparse representation of the manifold. While many embedding techniques exist, we require one that is fast and can be incrementally modified. Recently random projection has been proposed as an efficient means to preserve the manifold

structure (Hegde et al., 2007; Freund et al., 2007). We build our algorithm upon the online version of the Random Projection Tree (Dasgupta and Freund, 2007, Appendix I). A Random Projection Tree (RPtree) is a tree data structure with desirable theoretical properties that asymptotically traces the manifold. The basic idea is simple: as points arrive sequentially, they are spatially sorted into the RPtree leaves. When enough points fall into a leaf, the RPtree grows by splitting the leaf along a hyperplane with random orientation. An RPtree can be regarded as an efficient online clustering algorithm whose clusters grow over time and cover the manifold, as shown in Figure 3.1. We refer the reader to Dasgupta and Freund (2007) for details, while presenting our extensions for semi-supervised learning below.

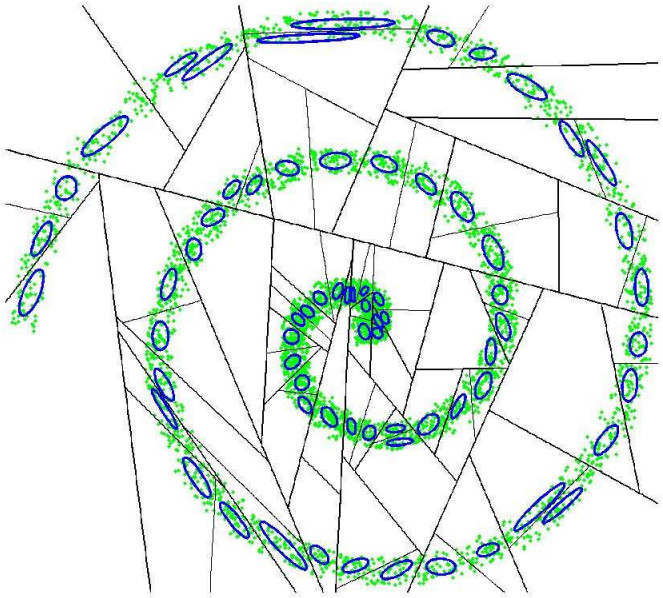


Figure 3.1: A random projection tree on the Swiss roll data. Small dots represent data points, line segments represent the random splits in the internal nodes of the RPtree, polygons represent the regions governed by the leaves, and ellipses represent the Gaussian distributions on the data points within each leaf. We exploit the fact that these distributions follow the manifold structure of the data.

Let $\{L_i\}_{i=1}^s, s \ll t$ denote the leaves in the RPtree at time t. To model the data points that have fallen into each leaf, we maintain a Gaussian distribution $\mathcal{N}(\mu_i, \Sigma_i)$ at each leaf L_i, where μ_i and Σ_i are estimated *incrementally* as the data points arrive. We also keep track of n_i, the number of points in leaf L_i. With an RPtree, we approximate the kernel representation of f_t (3.4) by the means of the Gaussian distributions associated with the tree leaves:

$$f_t = \sum_{i=1}^{s} \beta_i^{(t)} K(\mu_i, \cdot). \tag{3.10}$$

We approximate the instantaneous risk (3.2) by

$$J_t(f) \quad = \quad \frac{T}{l}\delta(y_t)c(f(x_t), y_t) + \frac{\lambda_1}{2}\|f\|_K^2 + \lambda_2 \sum_{i=1}^{s} n_i (f(\mu_i) - f(x_t))^2 w_{\mu_i t}. \tag{3.11}$$

From a graph regularization point of view, this can be understood as having a coarser graph over the RPtree leaves, and connecting the incoming point x_t to each leaf. We define the edge weight $w_{\mu_i t}$ between incoming point x_t and each leaf L_i to be

$$\begin{aligned} w_{\mu_i t} &= \mathbb{E}_{x \sim \mathcal{N}(\mu_i, \Sigma_i)} \left[\exp\left(-\frac{||x - x_t||^2}{2\sigma^2} \right) \right] \\ &= (2\pi)^{-\frac{d}{2}} |\Sigma_i|^{-\frac{1}{2}} |\Sigma_0|^{-\frac{1}{2}} |\tilde{\Sigma}_i|^{\frac{1}{2}} \\ &\quad \exp\left(-\frac{1}{2} \left(\mu_i^\top \Sigma_i^{-1} \mu_i + x_t^\top \Sigma_0^{-1} x_t - \tilde{\mu}_i^\top \tilde{\Sigma}_i \tilde{\mu}_i \right) \right), \end{aligned} \tag{3.12}$$

where $\Sigma_0 = \sigma^2 I$, $\tilde{\Sigma}_i = (\Sigma_i^{-1} + \Sigma_0^{-1})^{-1}$, $\tilde{\mu}_i = \Sigma_i^{-1}\mu_i + \Sigma_0^{-1} x_t$, and σ is the bandwidth of the (original point to point) weight. We call this weight scheme "**RPtree PPK**" for its similarity to the probability product kernel (Jebara et al., 2004). An even simpler approximation is to ignore the covariance structure by defining

$$w_{\mu_i t} \quad = \quad \exp\left(-\frac{||\mu_i - x_t||^2}{2\sigma^2} \right).$$

It has computational advantages at the price of precision. We call this weight scheme "**RPtree**."

With an RPtree, the function update occurs in three steps. In the first step, upon receiving x_t, we update f_t to an intermediate function f' using a basis of $s+1$ elements: $\mu_1, \ldots, \mu_s$ and x_t. This is similar to (3.8) in the buffering case. In the second step, the RPtree itself is adjusted to account for the addition of x_t. The adjustments include updating the Gaussian parameters for the

leaf that x_t falls into, and potentially splitting the leaf. In the latter case, the number of leaves s will increase to s', and each new leaf's mean and covariance statistics are established. In the third step, we approximate f' by f_{t+1} using the s' new basis elements $\mu_1, \ldots, \mu_{s'}$ ($s' = s$ if no split happened), similar to (3.9). The point x_t is then discarded.

3.4 Experiments

We present a series of experimental results as empirical evidence that online manifold regularization (MR) is a viable option for performing fast MR on large datasets. We summarize our findings as follows:

1. Online MR scales better than batch MR in time and space. Although recent advances in manifold regularization greatly improve the feasible problem size (Tsang and Kwok, 2006), we believe that it takes online learning to handle unlimited input sequences and achieve life-long learning.

2. Online MR achieves comparable performance to batch MR. This is measured by two criteria:

   (a) $J_{air}(T)$ approaches $J(f^*)$, both for the basic online MR algorithm, as well as for the buffering and RPtree approximations.

   (b) Generalization error of $\bar{f}$ approaches that of f^* on test sets.

3. Online MR can handle concept drift (changes in $P(x)$ and $P(y \mid x)$). The online method (using a limited size buffer) can track a non-stationary distribution and maintain good generalization accuracy, while the batch method trained on all previous data fails to do so.

Our focus is on comparing online MR to batch MR, not semi-supervised learning to supervised learning. It is known that semi-supervised learning does not necessarily outperform supervised learning, depending on the correctness of model assumptions. Thus, our experiments use tasks where batch MR has proven beneficial in prior work, and we demonstrate that online MR provides a useful alternative to batch MR on these tasks.

3.4.1 Datasets and Protocol

We report results on three datasets. The first is a toy two-spirals dataset. The training sequences and test sets (of size 2000) are generated i.i.d. . The second is the MNIST digit classification dataset (LeCun et al., 1998), and we focus on two binary tasks: 0 vs. 1 and 1 vs. 2. We scaled down the images to 16 x 16 pixels (256 features). The training sequences are randomly shuffled subsets of the official training sets, and we use the official test sets (of size 2115 for 0 vs. 1, and 2167 for 1 vs. 2). The third is the 361-dimensional Extended MIT face vs. non-face image classification dataset ("Face") (Tsang et al., 2005). We sampled a balanced subset of the data, and split this into a training set and a test set. The same test set of size 2000 is used in all experiments, while different training runs use different randomly shuffled subsets of the training set. The labeled rate p_l is 0.02 in all experiments, with points assigned to each class with equal probability.

Our experimental protocol is the following:

1. Generate randomly ordered training sequences and test sets (for MNIST and Face, the test sets are already given).

2. For batch MR, train separate versions on increasing subsequences (i.e., $T = 500, 1000, 2000, \ldots$).

3. For online MR, train once on the entire sequence.

4. For each T, compare the corresponding batch MR solution f^* with the online classifier trained up to T.

All results are the average of five such trials. The error bars are ± 1 standard deviation.

All experiments use hinge loss c and RBF kernel K. The kernel bandwidth parameter σ_K, λ_1, λ_2, and the edge weight parameter σ were all tuned for batch MR using $T = 500$. When using a limited size buffer, we set $\tau = 300$, and only require that matching pursuit reduce the residue norm by 50%. We use a step size of $\eta_t = \gamma/\sqrt{t}$, where $\gamma = 0.03$ for the RPtree approximation, and 0.1 for all other methods. We implemented all methods using MATLAB and CPLEX.

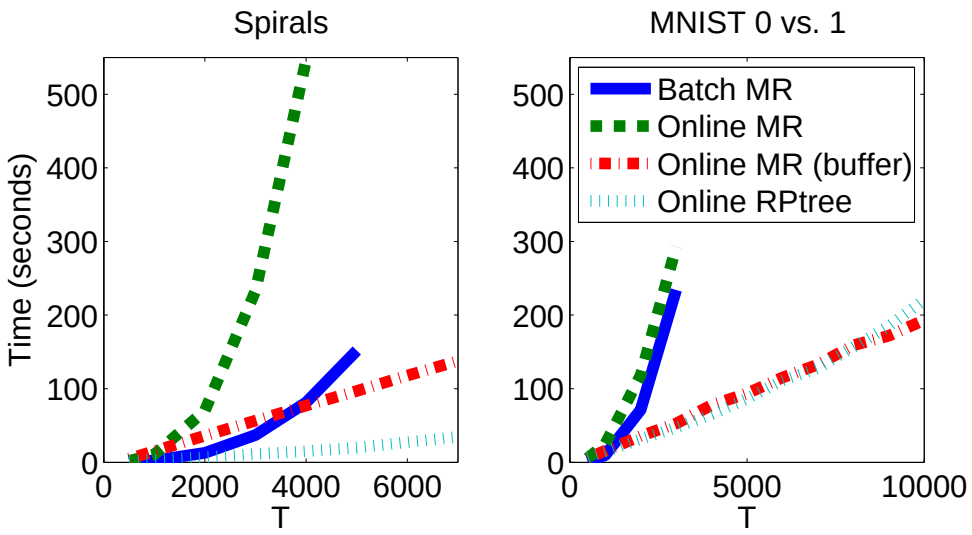


Figure 3.2: Runtime growth curves. Batch MR and basic online MR scale quadratically, while the sparse approximations of buffering and RPtree scale only linearly. Batch MR runs out of memory after $T = 5000$, and we stop basic online MR at $T = 4000$ because the runtime becomes excessive. Though not shown, online RPtree PPK has a curve nearly identical to online MR (buffered).

3.4.2 Online MR Scales Better than Batch MR

We illustrate this point by comparing runtime growth curves on the spirals and MNIST 0 vs. 1 datasets. Figure 3.2(left) shows that, for the spirals dataset, the growth rates of batch MR and basic online MR are quadratic as expected (in fact, online MR has more overhead in our MATLAB implementation). Batch MR runs out of memory after $T = 5000$, and we stop basic online MR at $T = 4000$ because the runtime becomes excessive. On the other hand, online MR (buffered) and online RPtree are linear. Though not included in the plot, online RPtree PPK has a curve nearly identical to online MR (buffered). Figure 3.2(right) demonstrates similar trends for the higher dimensional MNIST 0 vs. 1 dataset.

3.4.3 Online MR Achieves Comparable Risks

We compare online MR's average instantaneous risk $J_{air}(T)$ vs. batch MR's risk $J(f^*)$ on the training sequence. Our experiments support the theory that $J_{air}(T)$ converges to $J(f^*)$ as

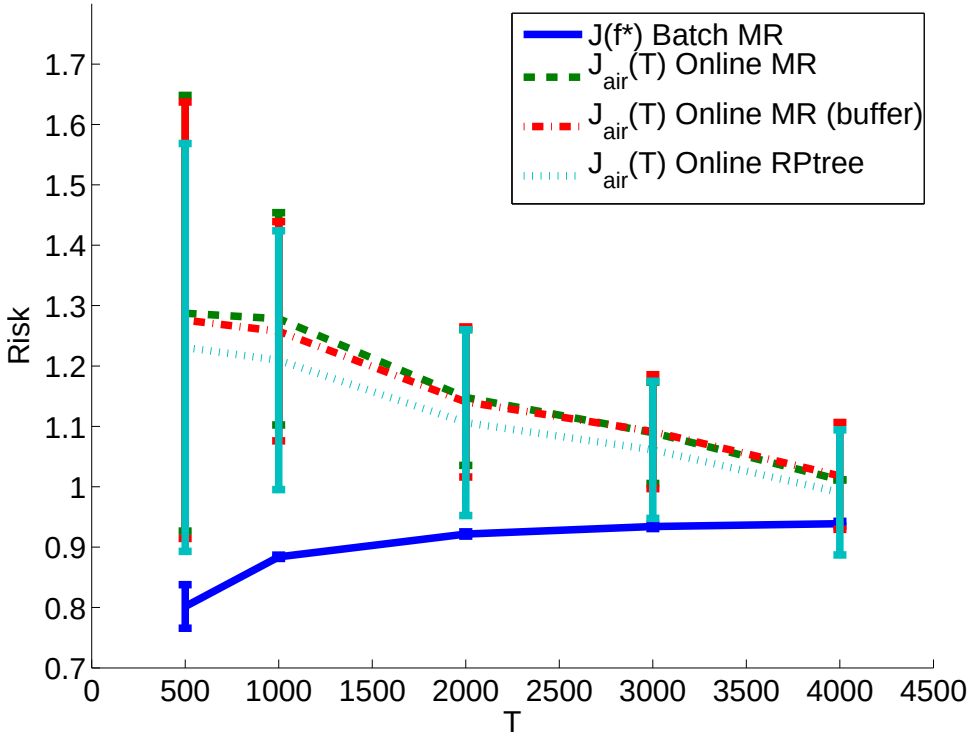


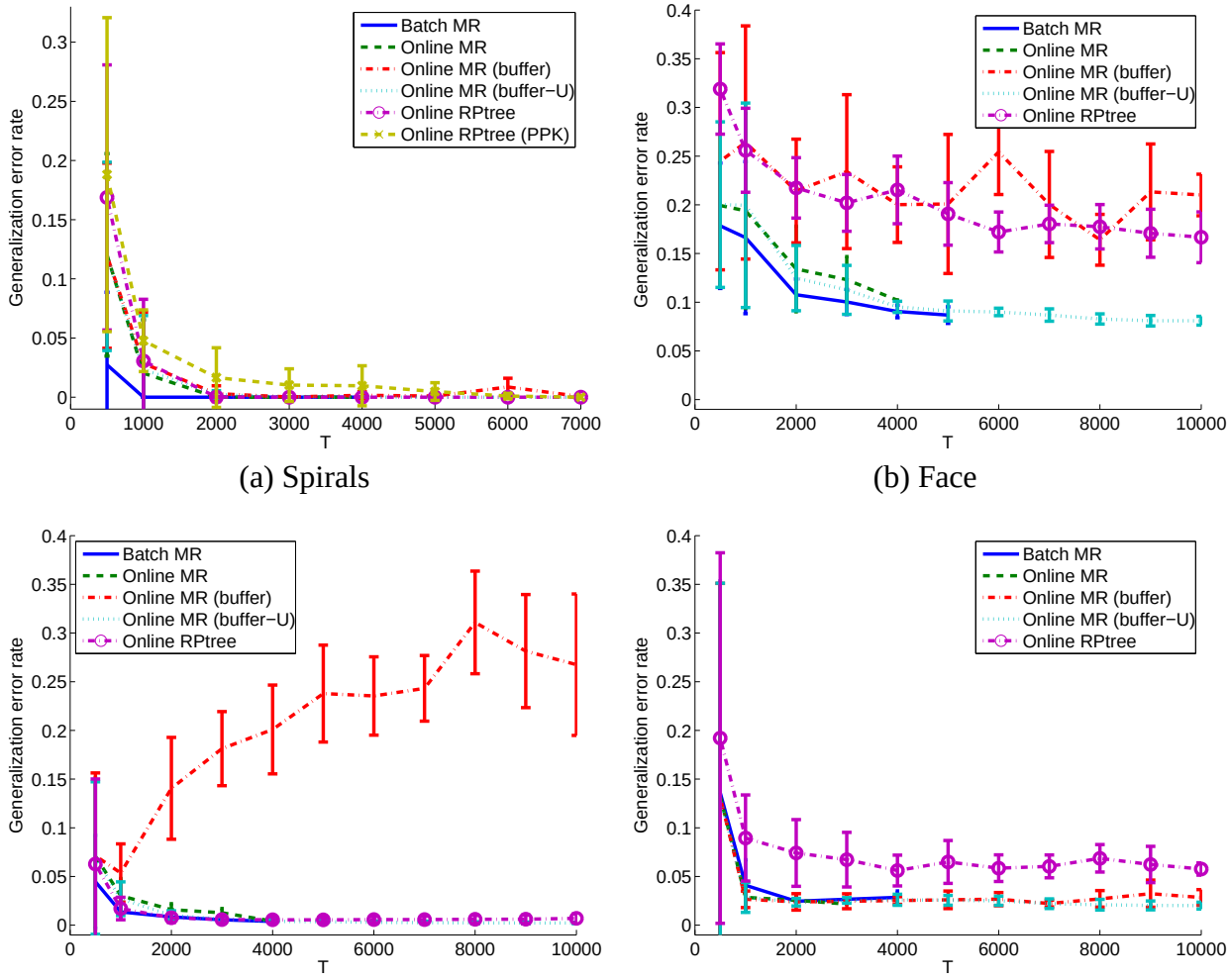
Figure 3.3: Online MR's average instantaneous risk $J_{air}(T)$ approaches batch MR's risk $J(f^*)$ as T increases.

T increases.[1] Figure 3.3 compares these measures for basic online MR and batch MR on the spirals dataset. The two curves approach each other. $J_{air}(T)$ continues to decrease beyond $T = 4000$ (not pictured). Figure 3.3 also shows that online MR (buffer) and online RPtree are good approximations to basic online MR in terms of J_{air}.

3.4.4 Generalization Error of Online MR

The experiments in this section compare the averaged function $\bar{f}$ of online MR and the batch solution f^* in terms of generalization error on test sets. Figure 3.4 presents results for all the

[1]While the average regret approaches zero asymptotically, the step size of $\eta_t = 1/\sqrt{t}$ decays rapidly, potentially leading to slow convergence. Thus, it is possible that long sequences (i.e., large T values) would be required for the online algorithm to compete with the best batch algorithm. Nevertheless, our experiments show this is not actually a problem in practice.



(a) Spirals (b) Face

(c) MNIST 0 vs. 1 (d) MNIST 1 vs. 2

Figure 3.4: Generalization error of batch MR's f^* and online MR's $\bar{f}$ as T increases. All the online methods perform nearly as well as batch MR. Online MR buffer-U consistently achieves test accuracy comparable to batch MR. Buffering and RPtree perform as well as basic online MR, showing little sign of approximation error. Panels (b), (c), and (d) reveal that buffer-U can be much better than buffer by preserving the larger weights on labeled points, which approximate the function better.

datasets. We observe that online MR buffer-U is the best and consistently achieves test accuracy that is comparable to batch MR.

From Figure 3.4(a), we observe that, for the spirals dataset, all the online methods perform nearly as well as batch MR. As is to be expected, batch MR makes the most efficient use of the

data and reaches 0 test error first, while the online methods require only a little additional data to reach this level (after all, standard incremental learning usually needs multiple passes over the training set). Buffering and RPtree perform as well as basic online MR, showing little sign of approximation error. Panels (b), (c), and (d) in Figure 3.4 show that buffer-U can be much better than buffer. This is understandable, since matching pursuit may provide a poor approximation to the contributions of the discarded data point. In high dimensional space, there may be few similar data points remaining in the small buffer, so much of the weight assigned to discarded points is lost. Under the buffer-U strategy, we alleviate this issue by preserving the larger weights on labeled points, which approximate the function better. RPtree PPK on these high dimensional datasets involves expensive inversion of (often singular) covariance matrices and is not included in the comparison. The performance of RPtree is no better than buffer-U.

3.4.5 Online MR Handles Concept Drift

Lastly, we demonstrate that online MR can handle concept drift. When the underlying distributions, both $P(x)$ and $P(y|x)$, change during the course of learning, using buffered online MR is extremely advantageous. For this experiment, we "spin" the two spirals dataset so that the spirals smoothly rotate 360° in every 4000 points (Figure 3.5). All points in the space will thus change their true labels during the sequence. We still provide only 2% of the labels to the algorithms. The test set for a given T consists of 2000 points drawn from the current underlying distribution.

For this experiment, we show the generalization error of batch MR's f^* vs. online MR (buffer)'s f_T, since the latest function is expected to track the changes in the data. Figure 3.5 illustrates that online MR (buffer) is able to adapt to the changing sequence and maintain a small error rate. In contrast, batch MR uses all data points, which now tend to conflict heavily (i.e., newer data from one class overlaps with older data from the other class). As expected, the single batch classifier f^* is inadequate for predicting such changing data.

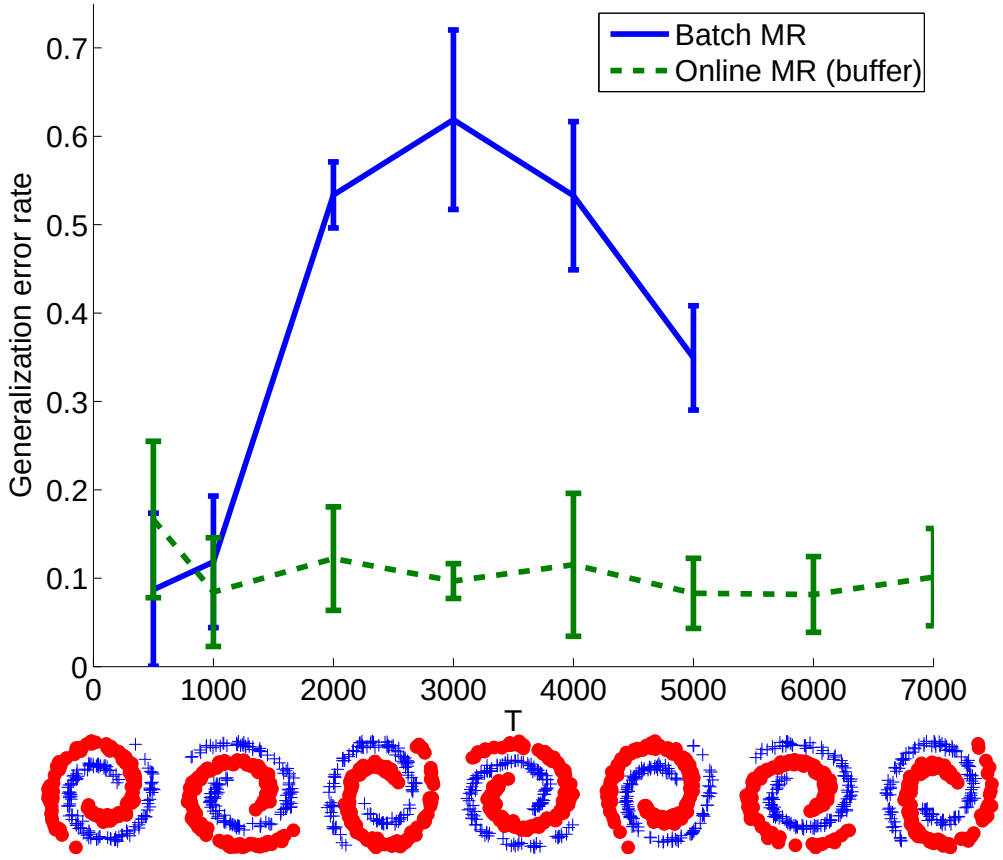


Figure 3.5: Online MR (buffer) has much better generalization error than batch MR when faced with concept drift in the rotating spirals dataset. Note that all points in the space change their true labels during the sequence, yet f_T is able to track the changes in the data. The single batch classifier f^* is unable to cope with this situation.

3.5 Conclusions and Future Work

We presented an online semi-supervised learning algorithm that parallels manifold regularization. Our algorithm is based on online convex programming in RKHS. We proposed two sparse approximations using buffering and online random projection trees to make online MR practical. The original batch manifold regularization algorithm has time complexity at least $O(T^2)$; so does the online version without sparse approximation. In contrast, the RPtree approximation has complexity $O(T \log T)$, where each iteration requires $O(\log T)$ leaf lookups (the tree's height is

$O(\log T)$ because each leaf contains a constant maximum number of points). Buffering has complexity $O(T)$. Experiments show that our online MR algorithm has risk and generalization error comparable to batch MR, but scales much better. In particular, online MR (buffer-U) tends to have the best performance.

While Chapter 4 presents an alternative approach to online semi-supervised learning, many interesting questions remain for the online SSL framework introduced in this chapter. On the practical side, one avenue for future work is to explore new strategies for maintaining a fixed-size memory budget while still adequately representing the marginal data distribution. For instance, we can explore many other sequential or incremental clustering algorithms in place of random projection trees. One concrete idea is to use a non-parametric Bayesian approach (e.g., Dirichlet Process Mixture Models (DPMM) (Neal, 2000)) to model the arriving data as a growing number of clusters. Though not an online method, incremental versions have been developed (Gomes et al., 2008; Zhang et al., 2004). The hope is that improvements in the clustering approximation will translate to improved predictions.

On the theoretical side, we can investigate different regret notions that might be appropriate for the online semi-supervised setting, performance guarantees with concept drift, and models that do not require all previous points. We may be able to leverage recent analysis of so-called "budget perceptrons," which cannot maintain all support vectors in memory (Sutskever, 2009). Two such (supervised) algorithms have formal performance guarantees: the Forgetron (Dekel et al., 2005), which replaces the oldest vector in a fixed-length buffer with a new vector on which the current classifier makes an error, and the Randomized Budget Perceptron (Cavallanti et al., 2007), which discards a random vector instead of the oldest.

The unified analysis of Sutskever (2009) views these algorithms as performing stochastic gradient descent with noisy and incorrect gradients. We can view our buffer-based online manifold regularization algorithm, which operates on a fixed-size dynamic graph, as using a noisy gradient of the manifold regularization risk. That is, at each step we compute $\nabla_t = \frac{\partial J_t(f)}{\partial f}$, where $J_t(f)$ is defined in (3.2), while really we wish to move along the gradient of the batch risk in (3.1): $\nabla = \frac{\partial J(f)}{\partial f}$. Our goal is to prove that the gradient with respect to the dynamic graph equals the

true gradient in expectation (over random data): $\mathbb{E}[\nabla_t] = \nabla$. If we can bound the error in the gradient when the graph is constructed in a particular way (which may be different than the current scheme), then we can apply an approach similar to that of Sutskever (2009) to establish a formal guarantee for the framework presented in this chapter.

Chapter 4

OASIS: Online Active SemI-Supervised Learning

In this chapter,[1] we continue to examine online or incremental SSL as an alternative to the batch approach; labeled and unlabeled data are processed one at a time, predictions can be made at any time, and only a bounded amount of storage is needed to handle an unlimited stream of data. Furthermore, it is natural to incorporate active learning (Settles, 2009) in this setting; upon receiving an unlabeled data point, the learner may request the label from an oracle. Many real-world learning tasks fit nicely into this framework, such as classifying images collected by a surveillance camera, or categorizing blog posts and tweets in real-time as they emerge on the social Web. We present a novel, fully Bayesian algorithm capable of *online active semi-supervised learning (OASIS)*.

We consider the online SSL setting introduced in Chapter 3, and extend it to include an (optional) active learning component:

1. At time t, the world picks $\mathbf{x}_t \in \mathbb{R}^d$ and $y_t \in \{-1, 1\}$ and presents $\mathbf{x}_t$ to the learner.
2. The learner makes a prediction $\widehat{y}_t$ using its current model.
3. With a small probability p_l, the world reveals the label y_t.
4. If y_t is not revealed, the learner may choose to ask for it. Otherwise, $\mathbf{x}_t$ remains unlabeled.
5. The learner updates its model based on $\mathbf{x}_t$ and, if available, the label y_t.

This setting differs dramatically from traditional online learning where all data points are labeled. It also differs from the batch SSL setting where methods must wait to collect all the labeled and

[1] Based on joint work with Xiaojin Zhu, Alex Furger, and Junming Xu.

unlabeled data before beginning to learn and make semi-supervised predictions. The goal is for the model's predictions $\widehat{y}_t$ to be accurate; performance will be measured by the cumulative number of mistakes made by the learner.

Semi-supervised learning is often possible through some assumption about the interaction between the marginal data distribution $P(\mathbf{x})$ and the conditional label distribution $P(y \mid \mathbf{x})$. We have already discussed online SSL based on the manifold or graph-based assumption (Chapter 3). In the current chapter, we focus on the so-called *cluster* or *low-density gap assumption*, which states that the decision boundary induced by $P(y \mid \mathbf{x})$ ought to lie in a region of low data density (Chapelle and Zien, 2005). As discussed in Chapter 2, this assumption is at the heart of transductive or semi-supervised support vector machines (TSVMs or S3VMs) (Chapelle et al., 2008), as well as the Bayesian analog of null category noise model Gaussian Processes (Lawrence and Jordan, 2005). Locating a low-density gap is typically formulated as a non-convex optimization problem, which may miss other gaps in the data. Furthermore, even though some S3VMs are scalable (see Chapelle et al. (2008) for a review), they are batch algorithms and not suitable for life-long online learning with potentially unlimited amounts of data.

Our main contribution is a fully Bayesian approach to the low-density gap assumption in an online setting. We employ sequential Monte Carlo to efficiently track the posterior over the hypothesis space. By restricting the amount of data stored over time, we achieve constant time and space complexity per time step. Furthermore, maintaining the posterior leads to a principled active learning scoring method.

4.1 OASIS: Online Active SemI-Supervised Learning

4.1.1 Bayesian Model for the Gap Assumption

Recall that we observe a partially labeled sequence of feature vectors $\mathbf{x}_1, \mathbf{x}_2, \ldots$, where each $\mathbf{x}_t \in \mathbb{R}^d$. Let $D_t = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_t, y_t)\}$ be all the data observed through time t, where we use $y_t = 0$ for unlabeled data. Our goal is to learn a classifier to predict the class label of each incoming unlabeled data point, and then update the classifier based on the information we obtain ($\mathbf{x}_t$ alone or

$\mathbf{x}_t$,y_t).[2] Let us assume the classifier is parameterized by a weight vector $\mathbf{w} \in \mathbb{R}^d$, which interacts with the data through a linear function $f(x) = \mathbf{w}^\top \mathbf{x}$.[3] Throughout, we use the terms classifier and weight vector interchangeably. To define our Bayesian model, we begin by introducing a likelihood function that is sensitive to unlabeled data. Inspired by the null category noise model of Lawrence and Jordan (2005), in addition to the positive and negative classes, we model a third "null category" which is never actually observed, but occupies some region of probability mass in the likelihood function, acting as a Bayesian analog to an SVM's margin.[4] Our likelihood (visualized in Figure 4.1(a)) is defined as follows

$$p(y \mid \mathbf{x}, \mathbf{w}) = \begin{cases} \min(1-\gamma, \max(\gamma\alpha, c\exp(-yc'\mathbf{w}^\top \mathbf{x}))) & \text{if } y = +1 \text{ or } y = -1 \\ 1 - [p(y=+1 \mid \mathbf{x}, \mathbf{w}) + p(y=-1 \mid \mathbf{x}, \mathbf{w})] & \text{if } y = \emptyset \text{ (null category)} \end{cases}, \tag{4.1}$$

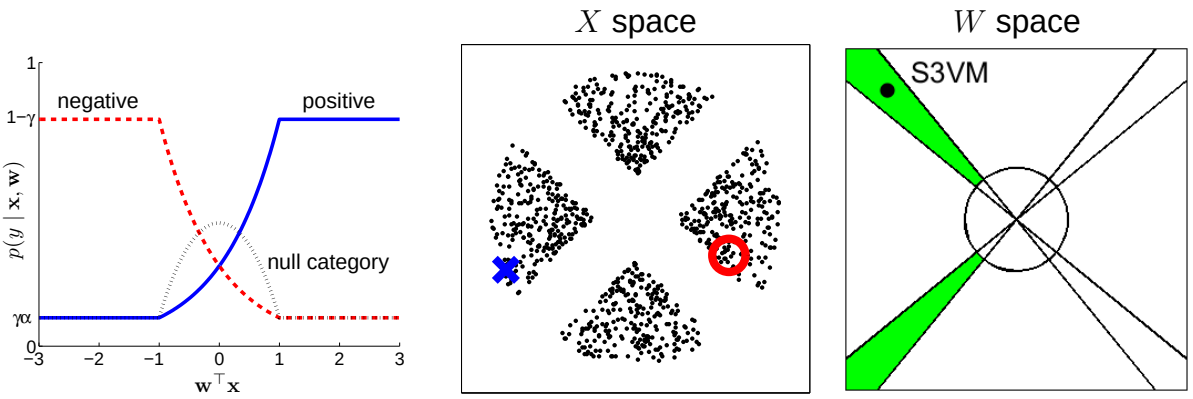
where $\gamma = 0.2$, $\alpha = 0.5$, $c = \sqrt{(1-\gamma)\gamma\alpha}$, $c' = \log(\frac{\gamma\alpha}{c})$, though other function forms leading to the same general shape would serve our purpose, too.

This likelihood has several interesting properties, which implement the gap assumption. It is flat beyond a margin value ($\mathbf{w}^\top \mathbf{x}$) of 1 or -1 to ensure that weight vectors placing data outside the margin are treated equally. Furthermore, due to the convex curvature of the positive and negative class likelihoods within $[-1, 1]$, there is a concave null category probability between -1 and 1. We never receive data from the null category; rather, unlabeled data will be considered being in either the positive or the negative class: $p(y \text{ unlabeled } \mid \mathbf{x}, \mathbf{w}) = p(y \in \{-1, 1\} \mid \mathbf{x}, \mathbf{w}) = p(y = +1 \mid \mathbf{x}, \mathbf{w}) + p(y = -1 \mid \mathbf{x}, \mathbf{w})$. Therefore, a weight vector $\mathbf{w}$ that places unlabeled data $\mathbf{x}$ in the "null category region" $\mathbf{w}^\top \mathbf{x} \in [-1, 1]$ has a low likelihood, which can always be increased by changing $\mathbf{w}$ to move $\mathbf{w}^\top \mathbf{x}$ toward -1 or 1. As a result, this likelihood favors decision boundaries that fall in a low-density region of the input space.

[2] For now, we assume a linear classifier, but the approach can be kernelized using the randomization trick of Rahimi and Recht (2007), as discussed in Section 4.3.

[3] In practice, we can handle a bias term by adding a dummy feature to all feature vectors.

[4] Despite the similar appearance of the likelihood functions, the current work is actually quite different from Lawrence and Jordan (2005); we are concerned with the strictly online setting, and we maintain the posterior over weight vectors through particle filtering, rather than making a Gaussian approximation which loses the critical ability to track multiple modes in the posterior. Such posterior is typical of semi-supervised learning.



(a) Likelihood (b) Dataset with 2 gaps (c) Multimodal posterior

Figure 4.1: (a) "Null category" likelihood function to encourage low-density separation. (b) Example dataset where tracking the full posterior is beneficial over S3VM's point estimate. (c) The posterior distribution over weight vectors for the data in (b)—the green shaded cones contain all weight vectors that classify the labeled data correctly while placing the unlabeled data outside the null category region.

To complete the model, we must specify a prior on the parameter $\mathbf{w}$, such as a Gaussian prior $p(\mathbf{w}) = \mathcal{N}(\mathbf{w}; 0, \Sigma)$, or independent Cauchy priors on each dimension

$$p(\mathbf{w}) = \prod_{i=1}^{d} \text{Cauchy}(\mathbf{w}_i; 0, \nu).$$

As discussed in the experiments, we follow Gelman et al. (2008) and use this Cauchy prior combined with standardized data.

With the likelihood and prior defined, we can apply Bayes rule to derive the posterior over weight vectors (after observing past data D_{t-1}), and the predictive distribution:

$$p(\mathbf{w} \mid D_{t-1}) = \frac{\prod_{i=1}^{t-1} p(y_i \mid \mathbf{x}_i, \mathbf{w}) p(\mathbf{w})}{\int \prod_{i=1}^{t-1} p(y_i \mid \mathbf{x}_i, \mathbf{w}') p(\mathbf{w}') d\mathbf{w}'} \tag{4.2}$$

$$p(y \mid \mathbf{x}_t, D_{t-1}) = \int p(y \mid \mathbf{x}_t, \mathbf{w}') p(\mathbf{w}' \mid D_{t-1}) d\mathbf{w}'. \tag{4.3}$$

Since we never actually predict the null category, we are interested in the following conditional probability when $y \in \{-1, 1\}$ but is unobserved:

$$p(y \mid \mathbf{x}_t, D_{t-1}, y \in \{-1, 1\}) = \frac{p(y \mid \mathbf{x}_t, D_{t-1})}{p(y = -1 \mid \mathbf{x}_t, D_{t-1}) + p(y = 1 \mid \mathbf{x}_t, D_{t-1})}. \tag{4.4}$$

In general, it is not possible to compute this probability in closed form. The next section describes how we transform this into a tractable solution for online SSL.

To appreciate why maintaining the posterior is desirable, consider the example dataset in Figure 4.1(b) containing two gaps within the unit circle. With only two labeled data in opposite wedges, a decision boundary in either gap is feasible, and thus the posterior (green shaded region in Figure 4.1(c)) is multimodal. All vectors outside the circle and in the cone regions will have maximum likelihood given all possible data. The green shaded cones contain all weight vectors that classify the labeled data correctly while placing the unlabeled data outside the null category region. Inside the circle, the vectors have magnitude too small to place all data outside the margin (since $\|x\| \leq 1$). A batch S3VM (or an online version using gradient methods) will find only a point estimate in one of the modes of the posterior. The key to OASIS is to maintain and update an estimate of this posterior (green shaded region) as labeled and unlabeled data arrives.

4.1.2 Online SSL via Particle Filtering

Given the Bayesian model defined in the preceding section, our goal is to track the posterior. In theory, this is done by repeatedly applying Bayes rule. The integrals involved in using the full posterior are intractable, though, so we must resort to approximate methods. In particular, we use particle filtering with resample-move to reduce particle degeneracy (Gilks and Berzuini, 2001). The complete OASIS algorithm is summarized in Algorithm 5 and explained below.

Particle filtering is a sequential Monte Carlo technique designed for tracking and approximating distributions that are not amenable to analytical representation (Doucet et al., 2001). It relies on maintaining a sample of so-called particles to approximate the true distribution in question. We approximate the posterior distribution $p(\mathbf{w} \mid D_{t-1})$ by m weighted particles:

$$p(\mathbf{w} \mid D_{t-1}) \approx \sum_{i=1}^{m} w_i \delta(\mathbf{w} - \mathbf{w}^{(i)}),$$

Input: Number of particles m, Prior distribution $p(\mathbf{w})$, ESS threshold ESS_0,
Proposal distribution $q(\widehat{\mathbf{w}} \mid \mathbf{w})$, Active learning score threshold s_0

Sample m initial particles $\mathbf{w}_0^{(i)}$ (classifiers) from the prior $p(\mathbf{w})$.

Assign weights to particles $w_i = \frac{1}{m}, i = 1, \ldots, m$.

for $t = 1, \ldots$ **do**

Receive $\mathbf{x}_t$ and possibly y_t.

Active: If unlabeled, query for y_t if score$(\mathbf{x}_t) < s_0$ (see (4.6)).

if y_t *is available* **then** update particle weights $w_i = w_i p(y = y_t \mid \mathbf{x}_t, \mathbf{w}_{t-1}^{(i)})$.

else update particle weights $w_i = w_i p(y \in \{-1, 1\} \mid \mathbf{x}_t, \mathbf{w}_{t-1}^{(i)})$.

if $(\sum w_i)^2 / \sum w_i^2 < ESS_0$ **then**

Resample-Move:

$\{\widehat{\mathbf{w}}^{(i)}\}_{i=1}^m \leftarrow$ Systematic resampling

$\{\mathbf{w}_t^{(i)}\}_{i=1}^m \leftarrow$ Metropolis-Hastings for each $\widehat{\mathbf{w}}^{(i)}$ (using proposal distribution q).

Reset particle weights $w_i = \frac{1}{m}, i = 1, \ldots, m$.

else

Keep existing (reweighted) particles: $\mathbf{w}_t^{(i)} = \mathbf{w}_{t-1}^{(i)}, i = 1, \ldots, m$.

Renormalize particle weights to sum to 1.

end

end

Algorithm 5: The OASIS algorithm for online, active semi-supervised learning.

where $\delta(\mathbf{w} - \mathbf{w}^{(i)}) = 1$ if $\mathbf{w} = \mathbf{w}^{(i)}$ and 0 otherwise. Each particle $\mathbf{w}^{(i)}, i = 1 \ldots m$ is a sample from this posterior and has an associated importance weight w_i. At time t, the predictive distribution can be approximated by particles as

$$P(y \mid \mathbf{x}_t, D_{t-1}) \approx \sum_{i=1}^{m} w_i p(y \mid \mathbf{x}_t, \mathbf{w}^{(i)}).$$

Recall, however, that the conditional probability

$$p(y \mid \mathbf{x}_t, D_{t-1}, y \in \{-1, 1\}) \approx \sum_{i=1}^{m} w_i \frac{p(y \mid \mathbf{x}_t, \mathbf{w}^{(i)})}{p(y = -1 \mid \mathbf{x}_t, \mathbf{w}^{(i)}) + p(y = 1 \mid \mathbf{x}_t, \mathbf{w}^{(i)})}$$

is used to make predictions for incoming data.

For online learning, at the beginning, we sample m particles from the prior and assign uniform initial weights $\frac{1}{m}$. Then, we repeatedly update the posterior based on the likelihood and the previous estimate of the posterior (which now acts as the prior). The new posterior distribution after observing $\mathbf{x}_t, y_t$ is proportional to $\sum_{i=1}^{m} w_i p(y_t \mid \mathbf{x}_t, \mathbf{w}^{(i)})\delta(\mathbf{w} - \mathbf{w}^{(i)})$. When a data point is observed, we update the posterior by reweighting the particles. From the above equation, we see that the new weight for $\mathbf{w}^{(i)}$ is obtained as the current weight multiplied by the likelihood $p(y_t \mid \mathbf{x}_t, \mathbf{w}^{(i)})$. If y_t is not observed, the weight is multiplied by $p(y_t \in \{-1, 1\} \mid \mathbf{x}_t, \mathbf{w}^{(i)}) = 1 - p(y_t = \emptyset \mid \mathbf{x}_t, \mathbf{w}^{(i)})$.

So far we have a basic method for incrementally updating an approximate posterior after observing new data. Classic particle methods, such as sampling importance resampling (SIR) (Doucet and Johansen, 2009), use the particle weights for resampling (with replacement), which results in a new generation of particles with weights reset to $\frac{1}{m}$. While theoretically justified, repeating this process many times is known to cause particle degeneracy—the number of distinct particles is non-increasing, so eventually few will remain. To minimize particle degeneracy, we apply the resample-move algorithm (Gilks and Berzuini, 2001), which provides a principled way to "jitter" particles and introduce diversity into the pool.

Resample-move consists of two steps. First, particles are resampled according to their weights—we apply the popular and effective "systematic resampling" (Doucet and Johansen, 2009). Then, each of the new particles is potentially moved to a nearby location. To ensure that the moved particles represent samples from the same posterior distribution as the old particles, we implement the move step using one step of the Metropolis-Hastings sampling algorithm (Metropolis et al., 1953; Hastings, 1970). We use a proposal distribution $q(\widehat{\mathbf{w}} \mid \mathbf{w})$ of the same form as the prior distribution, except centered on the starting particle location and with a smaller variance or scale. Using a symmetric proposal distribution allows us to compute the acceptance probability for each move using only the unnormalized posterior $f(\mathbf{w} \mid D_t)$:

$$\alpha(\widehat{\mathbf{w}}^{(i)}, \mathbf{w}^{(i)}) = \min\left(1, \frac{f(\widehat{\mathbf{w}}^{(i)} \mid D_t)}{f(\mathbf{w}^{(i)} \mid D_t)}\right), \tag{4.5}$$

where $\widehat{\mathbf{w}}^{(i)}$ is a proposed move, and $f(\mathbf{w} \mid D_t) = p(\mathbf{w}) \prod_{k=1}^{t} p(y_k \mid \mathbf{x}_k, \mathbf{w})$, where y_k is understood to be $\{-1, 1\}$ for unlabeled data.

4.1.3 Guaranteeing Bounded Time and Space Complexity Per Time Step

The astute reader will notice that computing (4.5) in the move step requires access to the entire history of data D_t, which is infeasible for learning on an unlimited stream of data. This is clearly undesirable for online learning and will quickly lead to a computational burden. Thus, we propose using an approximate Metropolis-Hastings step in which the acceptance probability is computed using only a fixed-length buffer of size τ. That is, we replace $f(\mathbf{w} \mid D_t)$ with

$$f(\mathbf{w} \mid D_t, \tau) = p(\mathbf{w}) \prod_{k=t-\tau+1}^{t} p(y_k \mid \mathbf{x}_k, \mathbf{w}).$$

While this approximation may result in periodically accepting moves that fall outside the true posterior, the method now satisfies our time and space goals and will be shown to be effective in practice. In addition, though not explored in the current work, using a τ-buffer can allow the method to handle concept drift by only relying on the most recent sample of data.

Even with a τ-buffer, computing the Metropolis-Hastings acceptance probability for each particle can be computationally intensive (though the runtime is constant per time step in the number of particles m). Therefore, as is customary in the literature (Doucet and Johansen, 2009; Ridgeway and Madigan, 2003), we only perform resample-move when the particles appear to show high redundancy, as measured by the so-called Effective Sample Size (ESS), which can be estimated by $(\sum w_i)^2 / \sum w_i^2$. If the ESS drops below a threshold ESS_0—typically $m/2$ (Doucet and Johansen, 2009), then we perform resample-move to improve diversity. Otherwise, we simply proceed to the next time step using the same set of particles, but reweighted.

4.1.4 Incorporating Active Learning

It is quite natural to incorporate active learning into the algorithm described thus far. The posterior can be viewed as a soft version space—the space of hypotheses consistent with the training data (Mitchell, 1997)—and like many active learning algorithms, we try to select queries that will

maximally pare down the version space. In our case, this translates to locating query points that will lead to downweighting and effectively killing off many particles.

To determine whether an incoming unlabeled item should be actively labeled, we first assign it a score based on the weighted average predictions made by the particles:

$$\text{score}(\mathbf{x}) = \left| \sum_{i=1}^{m} w_i \operatorname*{argmax}_{y \in \{-1,1\}} p(y \mid \mathbf{x}, \mathbf{w}^{(i)}) \right|. \tag{4.6}$$

This is the same disagreement-based score of Nowak (2009), which will be close to zero if roughly half of the particles predict positive and half negative, and the weights are close to uniform. Thus, querying the label of an item receiving a near-zero score will be very informative, as roughly half of the particles (the ones whose predictions disagree with the oracle label) will get downweighted. If the score is very large, then a clear majority vote exists, and we opt not to query.

While many schemes are possible to balance the trade-off between the cost of acquiring a label and the benefit of refining the model, we use a simple thresholding approach in the current work. Actively querying points that minimize the score criterion in (4.6) is theoretically justified in a pool-based active learning setting (Nowak, 2009). The same theory can be applied to the online active setting as well to justify a constant threshold. However, this analysis assumes unqueried points are ignored. Adapting the theory to account for the fact that we make updates based on unlabeled data between active queries remains an open issue for future work (see Section 4.3).

4.2 Empirical Evaluation

We conducted a series of experiments to compare OASIS to passive online semi-supervised and passive online supervised learning algorithms. We carefully tease apart OASIS's different elements and show that active online querying leads to better performance than random online labeling, in the context of online semi-supervised learning. Furthermore, the use of semi-supervised learning in the online setting (even without active querying) often outperforms the identical learner that ignores unlabeled data, as well as a state-of-the-art (supervised) online learner.

For all experiments, to avoid difficult parameter tuning under online semi-supervised conditions, we use the same prior and proposal distributions with a fixed set of default hyperparameters.

Following Gelman et al. (2008), which investigated default priors for Bayesian logistic regression models, we standardize the data such that each feature has mean 0 and standard deviation 0.5, and then place independent Cauchy$(0, 2.5)$ priors on each dimension. For the proposal distribution, we use a more peaked version of this distribution: Cauchy$(0, 0.025)$. Other parameters were fixed as follows: number of particles $m = 1000$, ESS threshold $ESS_0 = 500$, buffer size $\tau = 100$.

The experiments consider five algorithms:

- **[OASIS]**: Online, active, and semi-supervised (Algorithm 5).
- **[OSIS]**: Online and semi-supervised; no active learning, but otherwise same as OASIS.
- **[OS]**: Online and supervised; no active learning and ignores unlabeled data.
- **[AROW ($C = 1$)]**: State-of-the-art supervised "Adaptive Regularization of Weight Vectors" online learner (Crammer et al., 2009), run using code provided by the original authors with a default regularization parameter $C = 1$. This is a passive-aggressive, confidence-weighted classifier that maintains a diagonal-covariance Gaussian distribution over weight vectors.
- **[AROW (C^*)]**: We also report results for AROW using the per-trial optimal C in terms of total number of mistakes ("test-set tuned") to approximate supervised learning's mistake lower bound.

We use the following experimental procedure to compare active and passive algorithms, with and without the help of unlabeled data. Each experiment is based on 20 random trials of randomized sequences of T points. Each trial begins with $l = 2$ labeled examples (one per class, in random order), as we assume this is practical. While OASIS is the only algorithm under consideration that is able to actively query labels, we take care to ensure that each algorithm receives exactly the same number of total labels. Let a be the number of active queries OASIS makes on a given trial (a is some function of s_0 and the dataset). For each trial, we do the following: (i) Run OASIS with $l = 2$ initial labels and record a (number of queries); (ii) Run each other algorithm with $l = 2 + a$ labeled examples (first two, plus a randomly selected others). Note the same exact sequence of $\{\mathbf{x}_t\}$ vectors is used across the same trial for different algorithms. In this way, all

algorithms always see the same data and the same total number of labels; the algorithms differ in exactly which labels and how they deal with unlabeled data.

4.2.1 Synthetic Data

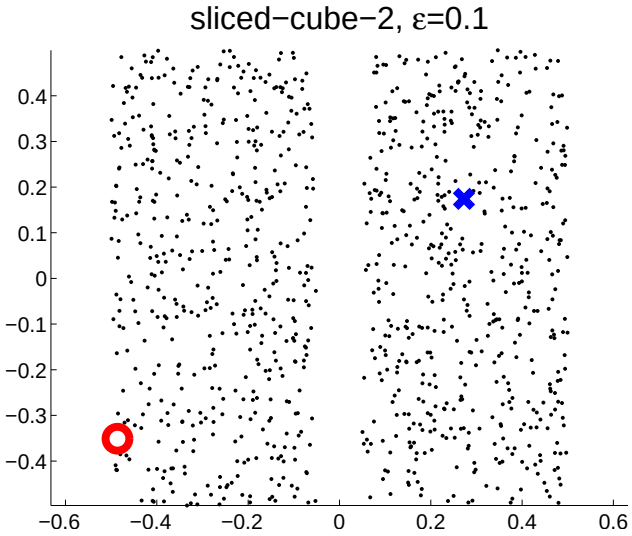
We begin by considering two synthetic datasets.

- **sliced-cube-d**: Uniformly distributed unit cube in $[-0.5, 0.5]^d$, with an ϵ-width slab removed from the first dimension to create two hyper-rectangles separated by a gap around the true decision boundary $x_1 = 0$ (Figure 4.2(a)).

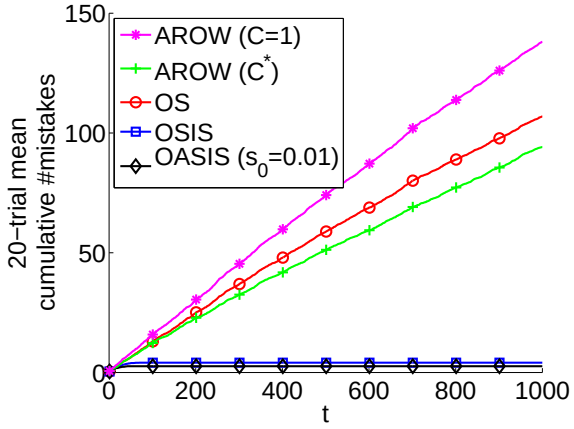
- **diced-cube-d**: Same as sliced-cube-d (with true decision boundary $x_1 = 0$), except ϵ-width slabs are removed from *all* dimensions to create 2^d hypercubes separated by potentially misleading low-density gaps (Figure 4.3(a)).

For both datasets, $\epsilon = (T/10)^{-1/d}$, such that the gaps should be large enough (relative to the average spacing between points) to be detectable after $T/10$ points (Singh et al., 2008).

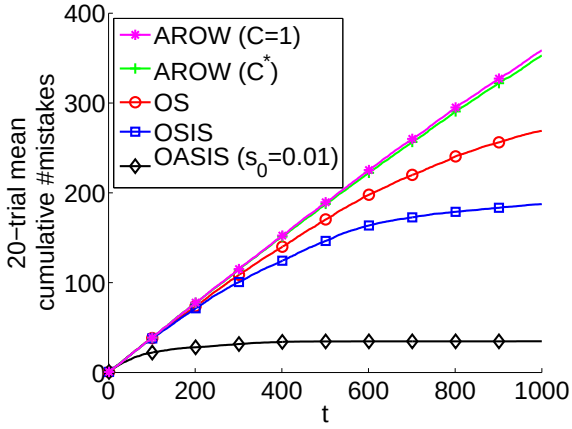
Figure 4.2(b,c) and Figure 4.3(b,c) plot the 20-trial average cumulative number of mistakes made by each algorithm when predicting the label of each incoming data point (regardless of whether the label ends up being revealed naturally or actively queried). The captions indicate the mean and standard deviation of a, the number of additional labels used in learning (via active selection for OASIS and random selection for the baselines). We observe that OASIS is able to very quickly learn the true decision boundary and stop making new mistakes across both datasets for all dimensionalities considered ($d \in \{2, 4, 8, 16, 32\}$, though only $d = 2$ and $d = 32$ are reported here). As expected, active querying allows OASIS to resolve ambiguities between the multiple gaps in the diced-cube-d datasets, though learning the decision boundary in this more confusing case takes longer on average. Comparing OSIS to OS and both versions of AROW, we see that SSL provides a large advantage, even when the few labeled data points are randomly selected. This example provides a proof of concept for the particle filtering approach to tracking the posterior, both in cases where the data distribution satisfies the gap assumption and when it contains misleading gaps.



(a) sliced-cube-2 data



(b) sliced-cube-2 (a=0.25 (0.44))

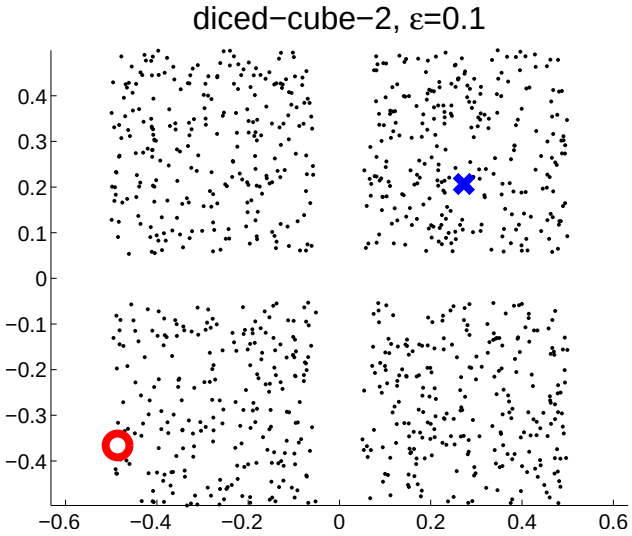


(c) sliced-cube-32 (a=4.6 (0.75))

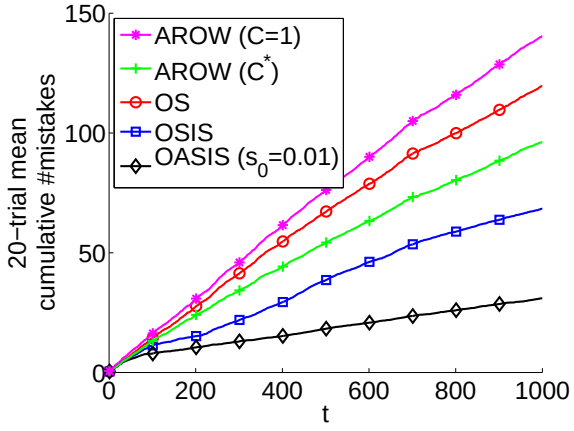
Figure 4.2: Sliced-cube-d synthetic data results for $T = 1000$, $l = 2$.

4.2.2 Real-World Data

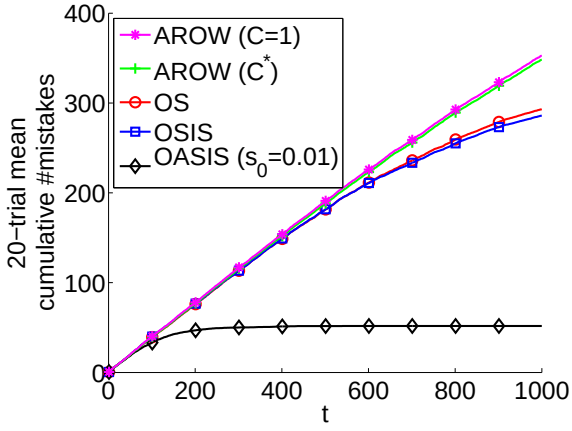
We next demonstrate that OASIS and its passive counterpart OSIS significantly outperform supervised baselines on real-world optical character recognition (OCR) tasks through their use of active sampling and online updates based on unlabeled data. We used two small scale datasets from the University of California, Irvine (UCI) machine learning repository, letter and pendigits,



(a) diced-cube-2 data



(b) diced-cube-2 ($a = 0.4(0.50)$)



(c) diced-cube-32 ($a = 4.85(0.75)$)

Figure 4.3: Diced-cube-d synthetic data results for $T = 1000$, $l = 2$.

in addition to the larger MNIST database. The details of the binary tasks we considered, and the results, are listed in Figure 4.4[5].

[5]For the MNIST data, we reduced the dimensionality down to 10 via "online PCA." To roughly simulate the online setting, principal components were found based on $\mathbf{x}_1, \ldots, \mathbf{x}_{1000}$, and $\mathbf{x}_{1001}, \ldots, \mathbf{x}_{10000}$ were simply projected into the resulting space.

On all three datasets, we see a clear ordering of performance: active SSL is better than passive SSL is better than passive supervised learning. We can measure statistically significant performance differences by applying two-sample t-tests to the total numbers of mistakes made by pairs of algorithms across the 20 trials[6]. On letter, OASIS significantly outperforms all the supervised algorithms ($p < 0.05$), and makes fewer mistakes than the passive semi-supervised OSIS. OSIS fails to achieve statistical significance at the 0.05 level over the supervised baselines, indicating that for this task, OASIS's active learning (rather than the use of unlabeled data) gives it the advantage. This suggests that perhaps the letter dataset's classes are not separated by a low density region. On pendigits and MNIST, though, both OASIS and OSIS make significantly fewer mistakes overall than each of the three supervised learners. Furthermore, on MNIST, OASIS significantly beats OSIS, demonstrating that actively queried labels can be more useful than randomly sampled labels in the context of online semi-supervised learning.

4.3 Conclusions and Future Work

We have presented a novel online learning algorithm, OASIS, which combines active learning and semi-supervised learning. OASIS exploits unlabeled data through the low-density gap assumption and is able to avoid the non-convex optimization typically associated with similar SSL algorithms by maintaining an approximation of the posterior over weight vectors via particle filtering. Outside of some special-purpose classifiers for computer vision tracking applications (Grabner et al., 2008; Tang et al., 2007), few authors have examined the task of online semi-supervised learning. We also include online active learning and show significant improvements over passive supervised baselines on both synthetic and real-world data.

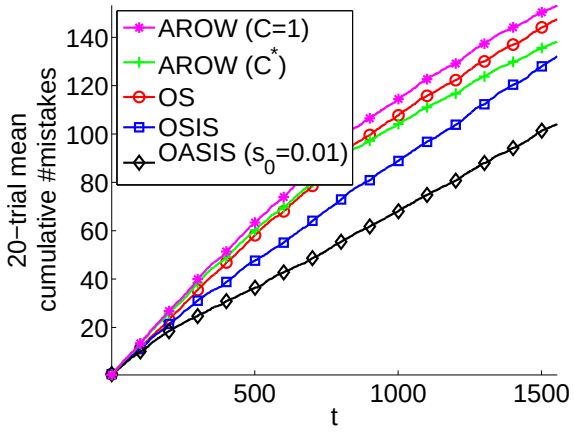
This chapter focuses on linear classifiers. It is possible, however, to incorporate kernels through the use of random features (Rahimi and Recht, 2007) and maintain the desirable effects of learning linear classifiers. This involves simply preprocessing the data by applying a set of random projections and then learning within the new "random feature" space.

[6]The specific labeled examples differ between OASIS and the passive algorithms, so the samples are not paired.

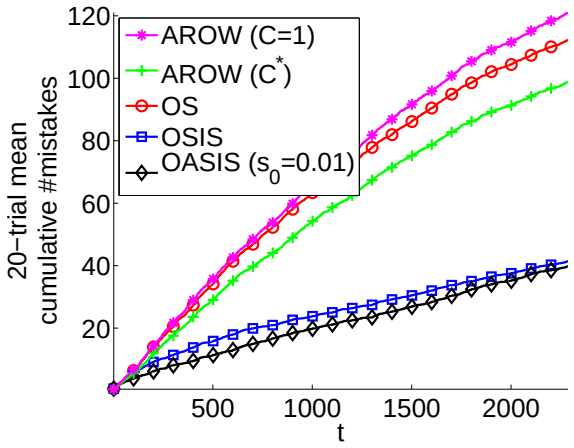
Our experiments focused on relatively low-dimensional datasets. It is well-known that particle filtering and sequential Monte Carlo techniques can be less efficient in high dimensions. One way to adapt OASIS to better handle much higher dimensional datasets, including those resulting from random-feature-based kernelization, is to improve the proposal distribution in the MCMC step of resample-move. The current approach uses a symmetric, peaky proposal distribution of the same family as the prior. This somewhat limits the ability of particles to jump between modes in the posterior. It is possible that we can achieve better mixing performance by convolving our current proposal distribution with a smoothed kernel density estimate based on the previous set of particles, thus allowing particles to jump to distant regions recently occupied by many particles.

Future work will also examine alternative active learning strategies, especially ones that strictly limit the frequency at which the learner can query, limit the total number of active queries, or consider costs associated with certain labels. The current fixed threshold strategy may not be suitable under all real-world constraints. While we could simply impose a hard limit on the number of queries, more sophisticated approaches may be possible which delicately balance the trade-off between receiving another label versus another unlabeled data point. In either case, we learn something, and it may be advantageous to delay querying until a more valuable point comes along. Many adaptive thresholding and selection criteria are possible for online active learning (see Beygelzimer et al. (2009) and the references therein), and careful modification to account for the role and impact of unlabeled data could lead to improved learning rates.

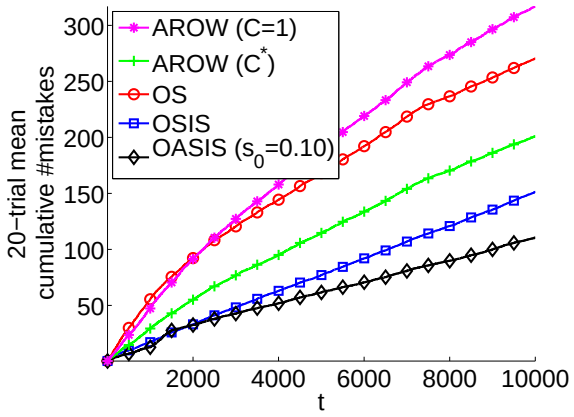
This concludes the discussion of novel online semi-supervised learning methods. The remainder of the dissertation focuses on expanding the reach of the more common batch setting to handle more complex datasets using novel assumptions based on unlabeled data.



(a) letter A vs B ($d = 16$), $T = 1555$, $l = 2$, $a = 5.10(1.92)$



(b) pendigits 0 vs 1 ($d = 16$), $T = 2286$, $l = 2$, $a = 2.60(1.14)$



(c) MNIST 0 vs 1 ($d = 10$), $T = 10000$, $l = 2$, $a = 10.30(5.01)$

Figure 4.4: Results on real-world OCR data.

Part III

Batch SSL: New Assumptions

Chapter 5

Multi-Manifold Semi-Supervised Learning

We now begin the first of several chapters introducing new assumptions for batch SSL. As discussed in Section 2.5, the graph used for graph-based semi-supervised learning plays a very important role. Having the wrong graph or poorly set weights can greatly impact performance. When the data lies on a mixture of manifolds, the standard graph types discussed in Section 2.5 may lead to diffusion of labels in undesirable ways. We try to remedy this problem in our work on multi-manifold semi-supervised learning (Goldberg et al., 2009). While expanding the reach of graph-based methods, this work is also a step in the direction of "safe SSL," as the theoretical analysis presented here suggests that our cluster-then-label procedure will be no worse than supervised learning under certain conditions.

The promising empirical success of semi-supervised learning algorithms in favorable situations has triggered several recent attempts (Balcan and Blum, 2005; Ben-David et al., 2008; Kaariainen, 2005; Lafferty and Wasserman, 2007; Niyogi, 2008; Rigollet, 2007) at developing a theoretical understanding of semi-supervised learning. For example, in a recent paper (Singh et al., 2008), it was established using a finite sample analysis that, if the complexity of the distributions under consideration is too high to be learnt using n labeled data points, but is small enough to be learnt using $m \gg n$ unlabeled data points, then semi-supervised learning (SSL) can improve the performance of a supervised learning (SL) task.

As discussed earlier, there have also been many successful practical SSL algorithms. However, the theoretical analyses and practical algorithms often assume that the data forms clusters or resides on a single manifold. Both a theory and an algorithm are lacking when the data is supported on a mixture of manifolds. Such data occurs naturally in practice. For instance, in

handwritten digit recognition, each digit forms its own manifold in the feature space; in computer vision motion segmentation, moving objects trace different trajectories which are low dimensional manifolds (Tron and Vidal, 2007). These manifolds may intersect or partially overlap, while having different dimensionality, orientation, and density. (See Figure 5.4 in the experiments section for some toy examples.) Existing SSL approaches cannot be directly applied to multi-manifold data. For instance, traditional graph-based SSL algorithms may create a graph that connects points on different manifolds near a manifold intersection, thus diffusing information across the wrong manifolds.

The main contributions of this work are:

- We generalize the theoretical analysis of Singh et al. (2008) to the case where the data is supported on a mixture of manifolds. We give a finite sample analysis to quantify the potential gain of using unlabeled data in this multi-manifold setting.
- Guided by the theory, we propose an SSL algorithm that handles multiple manifolds as well as clusters. It works by separating different manifolds into decision sets and performing supervised learning within each set.
- The algorithm builds upon novel Hellinger-distance-based graphs and size-constrained manifold clustering.
- Experiments show that our algorithm can perform SSL on multiple intersecting, overlapping, and noisy manifolds.

5.1 Theoretic Perspectives on Multi-Manifold Semi-Supervised Learning

In this section, we briefly review the conclusions of Singh et al. (2008), which are based on the cluster assumption, and then describe our new analysis for the single manifold and multi-manifold case (Goldberg et al., 2009).

The cluster assumption, as formulated by Singh et al. (2008), states that the target regression function or class label is locally smooth over certain subsets of the D-dimensional feature space

that are delineated by changes in the marginal density—throughout this work, we assume the marginal density is bounded above and below (away from zero). We refer to these delineated subsets as *decision sets*; i.e., all non-empty sets formed by intersections between the cluster support sets and their complements. If these decision sets, denoted by C, can be learnt using unlabeled data, the learning task on each decision set is simplified.

Previous results (Singh et al., 2008) suggest that if the decision sets can be resolved using unlabeled data, but not using labeled data, then semi-supervised learning can help. Singh et al. (2008) used finite sample bounds to characterize both the SSL gains and the relative value of unlabeled data.

To derive the finite sample bounds, the first step is to understand when the decision sets are resolvable using data. This depends on the interplay between the complexity of the class of distributions under consideration and the number of unlabeled points m and labeled points n. For the cluster case, the complexity of the distributions is determined by the margin γ, defined as the minimum separation between clusters or the minimum width of a decision set (Singh et al., 2008). If the margin γ is larger than the typical distance between the data points ($m^{-1/D}$ if using unlabeled data, or $n^{-1/D}$ if using only labeled data), then with high probability the decision sets can be learnt up to a high accuracy (which depends on m or n, respectively) (Singh et al., 2008). This implies that if $\gamma > m^{-1/D}$ (margin exists with respect to density of *unlabeled data*), then the finite sample performance (the expected excess error $\mathcal{E}rr$) of a semi-supervised learner $\widehat{f}_{m,n}$ relative to the performance of a clairvoyant supervised learner $\widehat{f}_{C,n}$, which has perfect knowledge of the decision sets C, can be characterized as follows:

$$\sup_{\mathcal{P}_{XY}(\gamma)} \mathcal{E}rr(\widehat{f}_{m,n}) \;\leq\; \sup_{\mathcal{P}_{XY}(\gamma)} \mathcal{E}rr(\widehat{f}_{C,n}) + \delta(m,n). \tag{5.1}$$

Here $\mathcal{P}_{XY}(\gamma)$ denotes the cluster-based class of distributions with complexity γ, and $\delta(m,n)$ is the error incurred due to inaccuracies in learning the decision sets using unlabeled data. Comparing this upper bound on the semi-supervised learning performance to a finite sample minimax lower bound on the performance of any supervised learner provides a sense of the relative performance

Complexity range	SSL upper bound	SL lower bound	SSL helps
Cluster Assumption			
$\gamma \geq n^{-\frac{1}{D}}$	$n^{-\frac{2\alpha}{2\alpha+D}}$	$n^{-\frac{2\alpha}{2\alpha+D}}$	No
$n^{-\frac{1}{D}} > \gamma \geq m^{-\frac{1}{D}}$	$n^{-\frac{2\alpha}{2\alpha+D}}$	$n^{-\frac{1}{D}}$	Yes
$m^{-\frac{1}{D}} > \gamma \geq -m^{-\frac{1}{D}}$	$n^{-\frac{1}{D}}$	$n^{-\frac{1}{D}}$	No
$-m^{-\frac{1}{D}} > \gamma$	$n^{-\frac{2\alpha}{2\alpha+D}}$	$n^{-\frac{1}{D}}$	Yes
Single Manifold $\kappa_{\text{SM}} := \min(r_0, s_0)$			
$\kappa_{\text{SM}} \geq n^{-\frac{1}{D}}$	$n^{-\frac{2\alpha}{2\alpha+d}}$	$n^{-\frac{2\alpha}{2\alpha+d}}$	No
$n^{-\frac{1}{D}} > \kappa_{\text{SM}} \geq m^{-\frac{1}{D}}$	$n^{-\frac{2\alpha}{2\alpha+d}}$	$\Omega(1)$	Yes
$m^{-\frac{1}{D}} > \kappa_{\text{SM}} \geq 0$	$O(1)$	$\Omega(1)$	No
Multi-Manifold $\kappa_{\text{MM}} := \text{sgn}(\gamma) \cdot \min(\lvert\gamma\rvert, r_0, s_0)$			
$\kappa_{\text{MM}} \geq n^{-\frac{1}{D}}$	$n^{-\frac{2\alpha}{2\alpha+d}}$	$n^{-\frac{2\alpha}{2\alpha+d}}$	No
$n^{-\frac{1}{D}} > \kappa_{\text{MM}} \geq m^{-\frac{1}{D}}$	$n^{-\frac{2\alpha}{2\alpha+d}}$	$\Omega(1)$	Yes
$m^{-\frac{1}{D}} > \kappa_{\text{MM}} \geq -m^{-\frac{1}{D}}$	$O(1)$	$\Omega(1)$	No
$-m^{-\frac{1}{D}} > \kappa_{\text{MM}}$	$n^{-\frac{2\alpha}{2\alpha+d}}$	$\Omega(1)$	Yes

Table 5.1: Conjectured finite sample performance of SSL and SL for regression of a Hölder-α, $\alpha > 1$, smooth function (with respect to geodesic distance in the manifold cases). These bounds hold for $D \geq 2$, $d < D$, $m \gg n$, and suppress constants and log factors.

of supervised learning (SL) vs. SSL. Thus, SSL helps if complexity of the class of distributions $\gamma > m^{-1/D}$ and *both* of the following conditions hold:

1. Knowledge of decision sets simplifies the supervised learning task, that is, the error of the clairvoyant learner $\sup_{\mathcal{P}_{XY}(\gamma)} \mathcal{E}rr(\widehat{f}_{C,n}) < \inf_{f_n} \sup_{\mathcal{P}_{XY}(\gamma)} \mathcal{E}rr(f_n)$, the smallest error that can be achieved by any supervised learner based on n labeled data. The difference quantifies the SSL performance gain.

2. m is large enough so that the error incurred due to using a finite amount of unlabeled data to learn the decision sets is negligible: $\delta(m,n) = O\left(\sup_{\mathcal{P}_{XY}(\gamma)} \mathcal{E}rr(\widehat{f}_{C,n})\right)$. This quantifies the relative value of labeled and unlabeled data.

The finite sample performance bounds on SSL and SL performance as derived in Singh et al. (2008) for the cluster assumption are summarized in Table 5.1 for the regression setting, where the target function is a Hölder-α smooth function on each decision set and $\alpha > 1$. We can see that SSL provides improved performance, by capitalizing on the local smoothness of the function on each decision set, when the separation between the clusters is large compared to the typical distance between unlabeled data $m^{-1/D}$ but less than the typical distance between labeled data $n^{-1/D}$. Negative γ refers to the case where the clusters are not separated, but can overlap and give rise to decision sets that are adjacent (see Singh et al. (2008)). In this case, SSL always outperforms SL provided the width of the resulting decision sets is detectable using unlabeled data. Thus, the interplay between the margin and the number of labeled and unlabeled data characterizes the relative performance of SL vs. SSL under the cluster assumption. Similar results can be derived in the classification setting where an exponential improvement (from $n^{-1/D}$ to e^{-n}) is possible provided the number of unlabeled data m grows exponentially with n (Singh et al., 2008).

5.1.1 Single Manifold Case

In the single manifold case, the assumption is that the target function lies on a lower d-dimensional manifold, where $d < D$, and is Hölder-α smooth ($\alpha > 1$) with respect to the geodesic distance on the manifold. Hence knowledge of the manifold, or equivalently the geodesic distances between all pairs of data points, can be gleaned using unlabeled data and reduces the dimensionality of the learning task.

In the case of distributions supported on a single manifold, the ability to learn the geodesic distances well, and hence the complexity κ_{SM} of the distributions, depends on two geometric properties of the manifold—its minimum radius of curvature r_0 and proximity to self-intersection s_0 (also known as branch separation) (Bernstein et al., 2000). If $\kappa_{\text{SM}} := \min(r_0, s_0)$ is larger than the typical distance between the data points ($m^{-1/D}$ with unlabeled data, or $n^{-1/D}$ with only labeled data), then with high probability the manifold structure is resolvable and geodesic distances can be learnt up to a high accuracy (which depends on m or n, respectively). This can be achieved by using shortest distance paths on an ϵ- or k-nearest neighbor graph to approximate the geodesic

distances (Bernstein et al., 2000). The use of approximate geodesic distances to learn the target function gives rise to an error-in-variable problem. Though the overall learning problem is now reduced to a lower-dimensional problem, we are now faced with two types of errors—the label noise and the error in the estimated distances. However, the error incurred in the final estimation due to errors in geodesic distances depends on m which is assumed to be much greater than n. Thus, the effect of the geodesic distance errors is negligible, compared to the error due to label noise, for m sufficiently large. This suggests that for the manifold case, if $\kappa_{\text{SM}} > m^{-1/D}$, then finite sample performance of semi-supervised learning can again be related to the performance of a clairvoyant supervised learner $\widehat{f}_{C,n}$ as in (5.1) above, since $\delta(m, n)$ is negligible for m sufficiently large.

Comparing this SSL performance bound to a finite sample minimax lower bound on the performance of any supervised learner indicates SSL's gain in the single manifold case and is summarized in Table 5.1. These are conjectured bounds based on the arguments above and similar arguments in Niyogi (2008). The SSL upper bound can be achieved using a learning procedure adaptive to both α and d, such as the method proposed in Bickel and Li (2007)[1]. The SL lower bounds follow from the results in Tsybakov (2004) and Niyogi (2008). SSL provides improved performance by capitalizing on the lower-dimensional structure of the manifold when the minimum radius of curvature and branch separation are large compared to the typical distance between unlabeled data $m^{-1/D}$, but small compared to the typical distance between labeled data $n^{-1/D}$.

5.1.2 Multi-Manifold Case

The multi-manifold case addresses the generic setting where the clusters are low-dimensional manifolds that possibly intersect or overlap. In this case, the target function is supported on multiple manifolds and can be piecewise smooth on each manifold. Thus, it is of interest to resolve the manifolds, as well as the subsets of each manifold where the decision label varies smoothly (that are characterized by changes in the marginal density). The analysis for this case is a combination of the cluster and single manifold case. The complexity of the multi-manifold class of distributions,

[1]Note, however, that the analysis in Bickel and Li (2007) considers the asymptotic performance of SL, whereas here we are studying the finite-sample performance of SSL.

denoted κ_{MM}, is governed by the minimum of the manifold curvatures, branch separations, and the separations and overlaps between distinct manifolds. For the regression setting, the conjectured finite sample minimax analysis is presented in Table 5.1.

These results indicate that when there is enough unlabeled data, but not enough labeled data, to handle the complexity of the class, then semi-supervised learning can help by adapting to both the intrinsic dimensionality and smoothness of the target function. Extensions of these results to the classification setting are straightforward, as discussed under the cluster assumption.

Notice that in all the above cases, the semi-supervised learning performance is never worse than the performance of any supervised learner. This is true under the assumption that the number of decision sets is finite. To guard against breaking up the problem into too many subproblems, we can restrict the number of decision sets to $\log n$. This implies that if the true number of decision sets is less than $\log n$, the above results are still valid except for an additional log factor, and if the true number of decision sets is more than $\log n$, then a performance gain is not achieved, however the performance is no worse than that of a supervised learner.

5.2 A Multi-Manifold Learning Algorithm

Guided by the theoretical analysis in the previous section, we propose a "cluster-then-label" type of SSL algorithm (see Algorithm 6). It consists of three main steps:

1. It uses the unlabeled data to form a small number of *decision sets*, on which the target function is assumed to be smooth. The decision sets are defined in the ambient space, not just on the labeled and unlabeled points.

2. The target function within a particular decision set is estimated using only labeled data that fall in that decision set, and using a supervised learner specified by the user.

3. A new test point is predicted by the target function in the decision set it falls into.

There have been several cluster-then-label approaches in the SSL literature. For example, the early work of Demiriz et al. (1999) modifies the objective of standard k-means clustering

algorithms to include a class impurity term. El-Yaniv and Gerzon (2005) enumerate all spectral clusterings of the unlabeled data with varying number of clusters, which together with labeled data induce a hypothesis space. They then select the best hypothesis based on an Occam's razor-type transductive bound. Some work in constrained clustering is also closely related to cluster-then-label from an SSL perspective (Basu et al., 2008). Compared to these approaches, our algorithm has two advantages:

1. It is supported by our SSL minimax theory;
2. It handles both overlapping clusters and intersecting manifolds by detecting changes in support, density, dimensionality or orientation.

Our algorithm is also different from the family of graph-regularized SSL approaches, such as manifold regularization (Belkin et al., 2006) and earlier variants (Joachims, 2003; Zhou et al., 2003; Zhu et al., 2003). They also depend on the manifold assumption that the target function varies smoothly on the manifold. In contrast,

1. Our algorithm is a *wrapper* method, which uses any user-specified supervised learner SL as a subroutine. This allows us to directly take advantage of advances in supervised learning without the need to derive new algorithms.
2. Our theory ensures that, even when the manifold assumption is wrong, our SSL performance bound is the same as that of the supervised learner (up to a $\log$ factor).

Finally, step 1 of our algorithm is an instance of manifold clustering. Recent advances on this topic include generalized principal component analysis (Vidal et al., 2008) and lossy coding (Ma et al., 2007) for mixtures of linear subspaces, multiscale manifold identification with algebraic multigrid (Kushnir et al., 2006), locally linear embedding plus spectral clustering (Polito and Perona, 2002), tensor voting (Mordohai and Medioni, 2005), spectral curvature clustering (Chen and Lerman, 2008), and the translated Poisson mixture model (Haro et al., 2008) for mixtures of non-linear manifolds. Our algorithm is unique in two ways. First, its use of Hellinger distance offers a new approach to detecting overlapping clusters and intersecting manifolds. Second, our decision sets have minimum size constraints, which we enforce by constrained k-means.

Given: n labeled examples and M unlabeled examples, and a supervised learner SL,

1. Use the unlabeled data to infer $k \sim O(\log(n))$ decision sets $\widehat{C}_i$:
   (a) Select a subset of $m < M$ unlabeled points
   (b) Form a graph on the $n + m$ labeled and unlabeled points, where the edge weights are computed from the Hellinger distance between local sample covariance matrices
   (c) Perform size-constrained spectral clustering to cut the graph into k parts, while keeping enough labeled and unlabeled points in each part
2. Use the labeled data in $\widehat{C}_i$ and the supervised learner SL to train $\widehat{f}_i$
3. For test point $x^* \in \widehat{C}_i$, predict $\widehat{f}_i(x^*)$.

Algorithm 6: The Multi-Manifold Semi-Supervised Learning Algorithm.

5.2.1 Hellinger Distance Graph

Let the labeled data be $\{(x_i, y_i)\}_{i=1}^n$, and the unlabeled data be $\{x_j\}_{j=1}^M$, where $M \gg n$. The building block of our algorithm is a *local sample covariance matrix*. For a point x, define $N(x)$ to be a small neighborhood around x in Euclidean space. Let Σ_x be the local sample covariance matrix at x:

$$\Sigma_x = \sum_{x' \in N(x)} (x' - \mu_x)(x' - \mu_x)^\top / (|N(x)| - 1), \tag{5.2}$$

where $\mu_x = \sum_{x' \in N(x)} x'/|N(x)|$ is the neighborhood mean. In our experiments, we let $|N(x)| \sim O(\log(M))$ so that the neighborhood size grows with unlabeled data size M. The covariance Σ_x captures the local geometry around x.

Our intuition is that points x_i, x_j on different manifolds or in regions with different density will have different local geometries. This intuition is captured by the Hellinger distance between their local sample covariance matrices Σ_i, Σ_j. The squared Hellinger distance is defined between two pdf's p, q: $H^2(p,q) = \frac{1}{2}\int \left(\sqrt{p(x)} - \sqrt{q(x)}\right)^2 \mathrm{d}x$. By setting $p(x) = \mathcal{N}(x; 0, \Sigma_i)$, i.e., a Gaussian with zero mean and covariance Σ_i, and similarly $q(x) = \mathcal{N}(x; 0, \Sigma_j)$, we extend the

definition of Hellinger distance to covariance matrices:

$$H(\Sigma_i, \Sigma_j) \equiv H\left(\mathcal{N}(x; 0, \Sigma_i), \mathcal{N}(x; 0, \Sigma_i)\right) = \sqrt{1 - 2^{D/2}|\Sigma_i|^{1/4}|\Sigma_j|^{1/4}/|\Sigma_i + \Sigma_j|^{1/2}}, \quad (5.3)$$

where D is the dimensionality of the ambient feature space. We will also call $H(\Sigma_i, \Sigma_j)$ the Hellinger distance between the two points x_i, x_j. When $\Sigma_i + \Sigma_j$ is rank deficient, H is computed in the subspace occupied by $\Sigma_i + \Sigma_j$. The Hellinger distance H is symmetric and in $[0, 1]$. H is small when the local geometry is similar, and large when there is significant difference in density, manifold dimensionality or orientation. Example 3D covariance matrices and their H values are shown in Figure 5.1.

It would seem natural to compute all pairwise Hellinger distances between our dataset of $n+M$ points to form a graph, and apply a graph-cut algorithm to separate multiple manifolds or clusters. However, if x_i and x_j are very close to each other, their local neighborhoods $N(x_i), N(x_j)$ will strongly overlap. Then, even if the two points are on different manifolds the Hellinger distance will be small, because their covariance matrices Σ_i, Σ_j will be similar. Therefore, we select a subset of $m \sim O\left(M/\log(M)\right)$ unlabeled points so that they are farther apart while still covering the whole dataset. This is done using a greedy procedure that begins by taking all n labeled points and then selects a subset of m unlabeled points to approximately cover the dataset. Each of these $n + m$ points has its local covariance Σ computed from the original full dataset. We then discard the $M - m$ unselected unlabeled points. Notice, however, that the number m of effective unlabeled data points is polynomially of the same order as the total number M of available unlabeled data points.

We can now define a sparse graph on the $n+m$ points. Each point x is connected by a weighted, undirected edge to $O(\log(n + m))$ of its nearest Mahalanobis neighbors chosen from the the set of $n+m$ points too. The choice of $O(\log(n+m))$ allows neighborhood size to grow with dataset size. Since we know the local geometry around x (captured by Σ_x), we "follow the manifold" by using the Mahalanobis distance as the local distance metric at x: $d_M^2(x, x') = (x - x')^\top \Sigma_x^{-1}(x - x')$. For example, a somewhat flat Σ_x will preferentially connect to neighbors in or near the same flat subspace. The graph edges are weighted using the standard RBF scheme, but with Hellinger distance: $w_{ij} = \exp\left(-H^2(\Sigma_i, \Sigma_j)/(2\sigma^2)\right)$. Figure 5.2(a) shows a small part of a synthetic "dollar

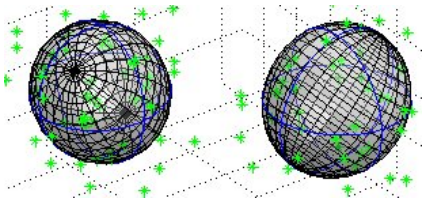
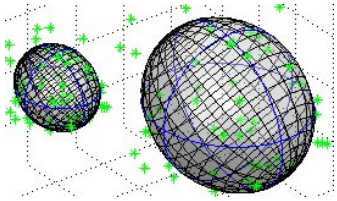
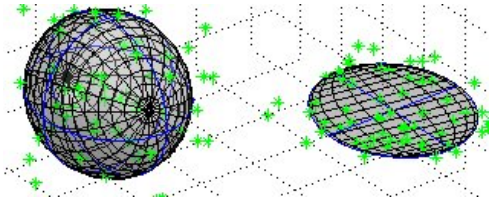
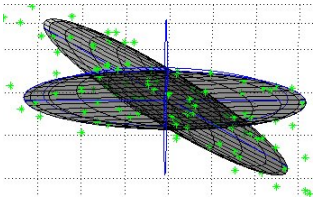
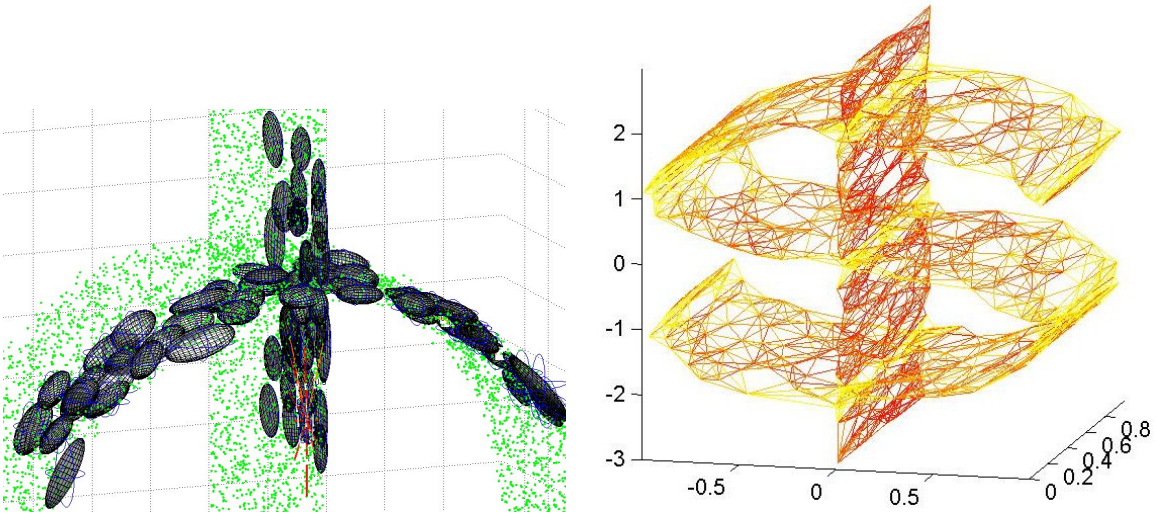
Covariance matrices	Comment	$H(\Sigma_1, \Sigma_2)$
	similar	0.02
	differ in density	0.28
	differ in dimensionality	1
	differ in orientation	1

Figure 5.1: Hellinger distance. Note that $H(\Sigma_1, \Sigma_2)$ is close to zero when the covariance matrices are similar. $H(\Sigma_1, \Sigma_2)$ is closer to (or exactly equal to) one, however, when the distributions in question differ in terms of density, dimensionality, or orientation.

sign" dataset, consisting of two intersecting manifolds: "S" and "|". The green dots are the original unlabeled points, and the ellipsoids are the contours of covariance matrices around a subset of



(a) (b)

Figure 5.2: The graph on the dollar sign dataset. (a) Subset of 3D covariance matrices centered on unlabeled points. (b) Complete graph with edge weights based on comparisons between nearby local covariance matrices. Darker red edges have large weights, while lighter yellow edges have small weights. Note that the darker edges tend to be within the same manifold.

selected unlabeled points within a small region. Figure 5.2(b) shows the graph on the complete dollar sign dataset, where red edges have large weights and yellow edges have small weights. Thus the graph combines locality and geometry: an edge has large weight when the two nodes are close in Mahalanobis distance, and have similar covariance structure.

5.2.2 Size-Constrained Spectral Clustering

We perform spectral clustering on this graph of $n + m$ nodes. We hope each resulting cluster represents a separate manifold, from which we will define a decision set. Of the many spectral clustering algorithms, we chose ratio cut for its simplicity, though others can be similarly adapted for use here. The standard ratio cut algorithm for k clusters has four steps (von Luxburg, 2007):

1. Compute the unnormalized graph Laplacian $L = Deg - W$, where $W = [w_{ij}]$ is the weight matrix, and $Deg_{ii} = \sum_j w_{ij}$ form the diagonal degree matrix.

2. Compute the k eigenvectors $v_1 \dots v_k$ of L with the smallest eigenvalues.

3. Form matrix V with $v_1 \dots v_k$ as columns. Use the ith row of V as the new representation of x_i.

4. Cluster all x under the new representation into k clusters using k-means.

Our ultimate goal of semi-supervised learning poses new challenges; we want our SSL algorithm to degrade gracefully, even when the manifold assumption does not hold. The SSL algorithm should not break the problem into too many subproblems and increase the complexity of the supervised learning task. This is achieved by requiring that the algorithm does not generate too many clusters and that each cluster contains "enough" labeled points. Because we will simply do supervised learning within each decision set, as long as the number of sets does not grow polynomially with n, the performance of our algorithm is guaranteed to be polynomially no worse than the performance of the supervised learner when the manifold assumption fails. Thus, we automatically revert to the supervised learning performance. One way to achieve this is to have three requirements:

1. The number of clusters grows as $k \sim O(\log(n))$.

2. Each cluster must have at least $a \sim O(n/\log^2(n))$ labeled points.

3. Each spectral cluster must have at least $b \sim O(m/\log^2(n))$ unlabeled points.

The first requirement sets the number of clusters k, allowing more clusters and thus handling more complex problems as labeled data size grows, while suffering only a logarithmic performance loss compared to a supervised learner if the manifold assumption fails. The second requirement ensures that each decision set has $O(n)$ labeled points up to log factor[2]. The third is similar, and makes spectral clustering more robust.

Spectral clustering with minimum size constraints a, b on each cluster is an open problem. Directly enforcing these constraints in graph partitioning leads to difficult integer programs (Ji,

[2]The square allows the size ratio between two clusters to be arbitrarily skewed as n grows. We do not want to fix the relative sizes of the decision sets *a priori*.

2004). Instead, we enforce the constraints in k-means (step 4) of spectral clustering. Our approach is a straightforward extension to the constrained k-means algorithm of Bradley et al. (2000). For point x_i, let $T_{i1} \dots T_{ik} \in \mathbb{R}$ be its cluster indicators: ideally, $T_{ih} = 1$ if x_i is in cluster h, and 0 otherwise. Let $c_1 \dots c_k \in \mathbb{R}^d$ denote the cluster centers. Constrained k-means is the iterative minimization over T and c of the following problem:

$$\begin{aligned} \min_{T,c} \quad & \textstyle\sum_{i=1}^{n+m} \sum_{h=1}^{k} T_{ih} \|x_i - c_h\|^2 \\ \text{s.t.} \quad & \textstyle\sum_{h=1}^{k} T_{ih} = 1,\ T \geq 0 \\ & \textstyle\sum_{i=1}^{n} T_{ih} \geq a,\ \sum_{i=n+1}^{n+m} T_{ih} \geq b,\ h = 1 \dots k, \end{aligned} \tag{5.4}$$

where we assume the points are ordered so that the first n points are labeled. Fixing T, optimizing over c is trivial, and amounts to moving the centers to the cluster means.

Bradley et al. (2000) showed that fixing c and optimizing T can be converted into a Minimum Cost Flow problem, which can be exactly solved. In a Minimum Cost Flow problem, there is a directed graph where each node is either a "supply node" with a number $r > 0$, or a "demand node" with $r < 0$. The arcs from $i \to j$ is associated with cost s_{ij}, and flow t_{ij}. The goal is to find the flow t such that supply meets demand at all nodes, while the cost is minimized:

$$\min_{t} \sum_{i \to j} s_{ij} t_{ij} \quad \text{s.t.} \sum_{j} t_{ij} - \sum_{j} t_{ji} = r_i,\ \forall i. \tag{5.5}$$

For our problem (5.4), the corresponding Minimum Cost Flow problem is shown in Figure 5.3. The supply nodes are $x_1 \dots x_{n+m}$ with $r = 1$. There are two sets of cluster center nodes. One set $c_1^\ell \dots c_k^\ell$, each with demand $r = -a$, is due to the labeled data size constraint. The other set $c_1^u \dots c_k^u$, each with demand $r = -b$, is due to the unlabeled data size constraint. Finally, a sink demand node with $r = -(n + m - ak - bk)$ catches all the remaining flow. The cost from x_i to c_h is $s_{ih} = \|x_i - c_h\|^2$, and from c_h to the sink is 0. It is then clear that the Minimum Cost Flow problem (5.5) is equivalent to (5.4) with $T_{ih} = t_{ih}$ and c fixed. Interestingly, (5.5) is proven to have integer solutions which correspond exactly to the desired cluster indicators.

Once size-constrained spectral clustering is completed, the $n + m$ points will each have a cluster index in $1 \dots k$. We define k decision sets $\{\widehat{C}_i\}_{i=1}^k$ by the Voronoi cells around these points:

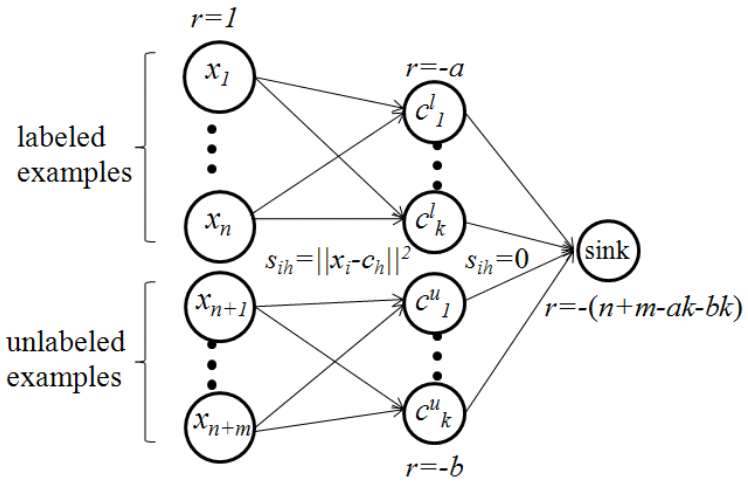


Figure 5.3: The Minimum Cost Flow problem equivalent to the step of constrained k-means clustering in which data points are reassigned to clusters (with cluster centers c fixed).

$\widehat{C}_i = \{x \in \mathbb{R}^D \mid x$'s Euclidean nearest neighbor among the $n+m$ points has cluster index $i\}$. We train a separate predictor $\widehat{f}_i$ for each decision set using the labeled points in that decision set, and a user-specified supervised learner. During test time, an unseen point $x^* \in \widehat{C}_i$ is predicted as $\widehat{f}_i(x^*)$. Therefore, the unlabeled data in our algorithm is used merely to determine the decision sets.

5.3 Experiments

We present experimental results showing that our algorithm consistently improves over SL in several different scenarios. In addition, we demonstrate that using our novel Hellinger-distance-based graph in the existing manifold regularization algorithm outperforms the same algorithm using a standard kNN graph.

5.3.1 Datasets

We experimented with five synthetic (Figure 5.4) and one real datasets. Datasets 1–3 are for regression, and 4–6 are for classification:

1. **Dollar sign** contains two intersecting manifolds. The "S" manifold has target y varying from 0 to 3π. The "|" manifold has target function $y = x_{\cdot 3} + 13$, where $x_{\cdot 3}$ is the vertical dimension. White noise $\epsilon \sim \mathcal{N}(0, 0.01^2)$ is added to y.

2. **Surface-sphere** slices a 2D surface through a solid ball. The ball has target function $y = ||x||$, and the surface has $y = x_{\cdot 2} - 5$.

3. **Density change** contains two overlapping rectangles. One rectangle is wide and sparse with $y = x_{\cdot 1}$, the other is narrow and five times as dense with $y = 10 - 5x_{\cdot 1}$. Together they produce three decision sets.

4. **Surface-helix** has a 1D toroidal helix intersecting a surface. Each manifold is a separate class.

5. **Martini** is a composition of five manifolds (classes) to form the shape of a martini glass with an olive on a toothpick, as shown in Figure 5.4(e).

6. **MNIST** digits. We scaled down the images to 16 x 16 pixels and used the official train/test split, with different numbers of labeled and unlabeled examples sampled from the training set.

5.3.2 Methodology & Implementation Details

In all experiments, we report results that are the average of 10 trials over random draws of M unlabeled and n labeled points. We compare three learners:

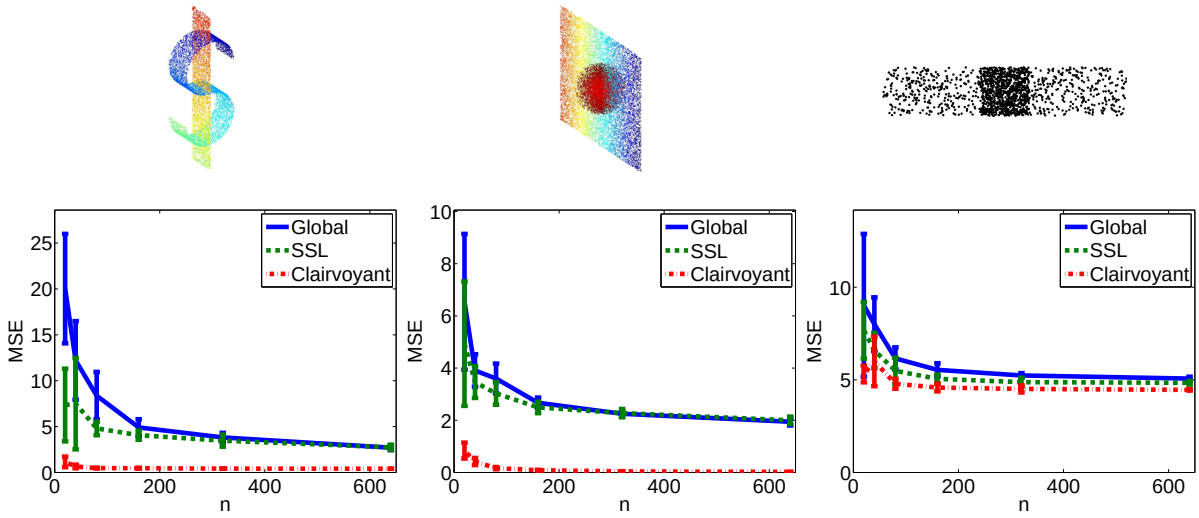
- **[Global]**: supervised learner trained on all of the labeled data, ignoring unlabeled data.
- **[Clairvoyant]**: with the knowledge of the true decision sets, trains one supervised learner per decision set.
- **[SSL]**: our semi-supervised learner that discovers the decision sets using unlabeled data, then trains one supervised learner per decision set.

After training, each classifier is evaluated on a massive test set, also sampled from the underlying distribution, to estimate generalization error. We implemented the algorithms in MATLAB, with Minimum Cost Flow solved by the network simplex method in CPLEX. We used the same set of parameters for all experiments and all datasets: We chose the number of decision sets to be $k = \lceil 0.5\ log(n) \rceil$. To obtain the subset of m unlabeled points, we let the neighborhood size $|N(x)| = \lfloor 3\, log(M) \rfloor$. When creating the graph W, we used $\lfloor 1.5\, log(m+n) \rfloor$ nearest Mahalanobis neighbors, and an RBF bandwidth $\sigma = 0.2$ to convert Hellinger distances to edge weights. The size constraints were $a = \lfloor 1.25n/log^2(n) \rfloor, b = \lfloor 1.25m/log^2(n) \rfloor$. Finally, to avoid poor local optima in spectral clustering, we ran 10 random restarts for constrained k-means, and chose the result with the lowest objective. For the regression tasks, we used kernel regression with an RBF kernel, and tuned the bandwidth parameter with 5-fold cross validation using only labeled data in each decision set (or globally for "Global"). For classification, we used a support vector machine (LIBSVM) with an RBF kernel, and tuned its bandwidth and regularization parameter with 5-fold cross validation. Note that LIBSVM solves multi-class problems using the 1-against-1 strategy. We used Euclidean distance in each decision region for the supervised learner, but we expect performance with geodesic distance would be even better.

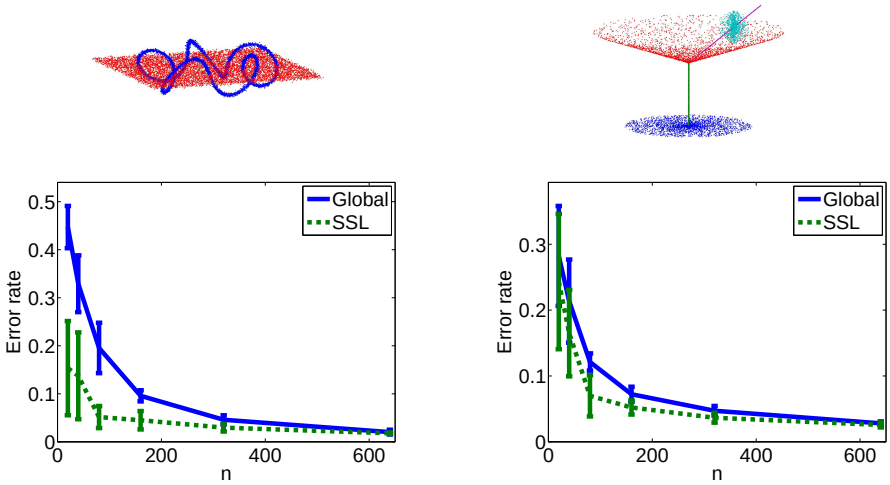
5.3.3 Results of Large M

Figure 5.4 reports the results for the five synthetic datasets. In all cases, we used $M = 20000$, $n \in \{20, 40, 80, 160, 320, 640\}$, and the resulting regressors/classifiers are evaluated in terms of MSE or error rate using a test set of 20000 points. These results show that our SSL algorithm can discover multiple manifolds and changes in density well enough to consistently outperform SL in both regression and classification settings of varying complexity. We also observed that even under- or over-partitioning into fewer or more decision sets than manifolds can still improve SSL performance[3].

[3]We compared Global and SSL's 10 trials at each n using two-tailed, paired t-tests. SSL was statistically significantly better ($\alpha = 0.05$) in the following cases: dollar sign at $n = 20$–80, density at $n = 40$–640, surface-helix at $n = 20$–320, and martini at $n = 40$–320. The two methods were statistically indistinguishable in other cases.



(a) Dollar sign (b) Surface-sphere (c) Density change



(d) Surface-helix (e) Martini

Figure 5.4: Regression MSE (a-c) and classification error (d-e) for synthetic datasets. All curves are based on $M = 20000$, 10-trial averages, and error bars plot ± 1 standard deviation. Clairvoyant classification error is 0.

We performed three experiments with the digit recognition data: binary classification of the digits 2 vs 3, and three-way classification of $1, 2, 3$ and $7, 8, 9$. Here, we fixed $n = 20$, $M = 5000$, 10 random training trials, each tested on the official test set. Table 5.2 contains results averaged

Method	2 vs 3	$1, 2, 3$	$7, 8, 9$
Global	0.17 ± 0.12	0.20 ± 0.10	0.33 ± 0.20
SSL	0.05 ± 0.01	0.10 ± 0.04	0.20 ± 0.10

Table 5.2: 10-trial average test set error rates $\pm$ one standard deviation for handwritten digit recognition with fixed $n = 20$ and $M = 5000$. All differences are statistically significant ($\alpha = 0.05$).

over these trials. SSL outperforms Global in all three digit tasks, and all differences are statistically significant ($\alpha = 0.05$). Note that we used the same parameters as the synthetic data experiments, which results in $k = 2$ decision sets for $n = 20$; again, the algorithm performs well even when there are fewer decision sets than classes. Close inspection revealed that our clustering step creates relatively pure decision sets. For the binary task, this leads to two trivial classification problems, and errors are due only to incorrect assignments of test points to decision sets. For the 3-way tasks, the algorithm creates 1+2 and 3 clusters, and 7+9 and 8 clusters. We conclude that the performance gains in the multi-class tasks are realized largely by decreasing errors on the 3 and 8 digits placed in their own decision set, while simplifying to two classes within the other decision set is also beneficial.

5.3.4 Effect of Too Small an M

Finally, we examine our SSL algorithm's performance with less unlabeled data. For the surface-helix dataset, we now fix $n = 80$ (which leads to $k = 3$ decision sets) and reduce M. Figure 5.5 depicts example partitionings for three M values, along with 10-trial average error rates ($\pm$ one standard deviation) in each setting. Note these are top-down views of the data in Figure 5.4(d). When M is small, the resulting subset of m unlabeled points is too small, and the partition boundaries cannot be reliably estimated. Segments of the helix shown in red and areas of the surface in blue or green correspond to such partitioning errors. Nevertheless, even when M is as small as 1000, SSL's performance is no worse than Global supervised learning, which achieves an error rate of 0.20 ± 0.05 when $n = 80$ (see Figure 5.4(d)).

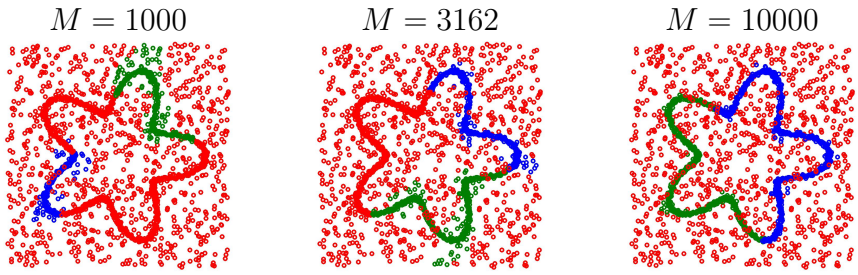


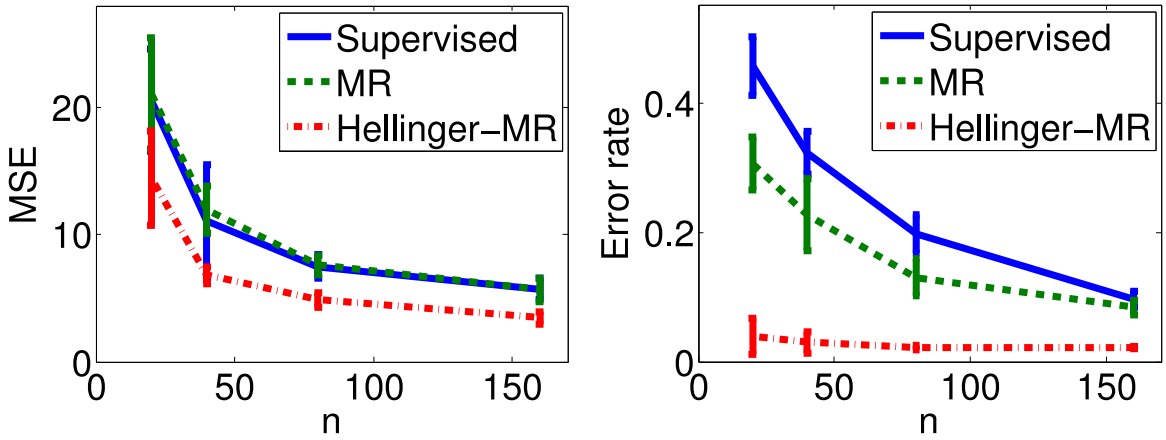
Figure 5.5: Effect of varying M for the surface-helix dataset ($n = 80$, which leads to $k = 3$ decision sets). Numbers listed are SSL's 10-trial average error rate $\pm$ one standard deviation. The images show top-down views of one trial's partitions of the data in Figure 5.4(d).

5.3.5 Manifold Regularization using the Hellinger Graph

The graph construction method presented here can be plugged into existing graph-based SSL algorithms. As seen in Figure 5.6, we found that manifold regularization (MR) (Belkin et al., 2006), using Euclidean kNN graphs with RBF weights and all parameters tuned using cross validation, performs worse than Global on these datasets due to the strong connections across manifolds. In contrast, replacing the kNN/RBF graph with the our Hellinger graph in the same regularization scheme (Hellinger-MR) leads to improved results.

5.4 Conclusions

We have extended SSL theory and practice to multi-manifolds. While we have quantified the theoretical performance of SSL vs SL in the single and multi-manifold case, a characterization of the relative value of unlabeled data requires a detailed analysis of how the inaccuracy in learning geodesic distances effects the learning error. A large scale empirical study on real datasets is also needed to demonstrate the robustness to the manifold assumption. These are subjects of future research.



(a) Dollar sign (b) Surface-helix

Figure 5.6: Comparison of Global/supervised learning, manifold regularization using a kNN/RBF graph (MR), and manifold regularization using our novel Hellinger graph (Hellinger-MR).

Chapter 6

Transduction with Matrix Completion: A Low-Rank Assumption for SSL

As discussed in several earlier chapters, semi-supervised learning methods make assumptions about how unlabeled data can help in the learning process, such as the manifold assumption (data lies on a low-dimensional manifold) and the cluster assumption (classes are separated by low density regions). In this chapter,[1] we present two transductive learning methods for handling classification problems with multiple labels per instance, based on the novel assumption that the feature-by-item and label-by-item matrices are *jointly low-rank*. This assumption effectively couples different label prediction tasks, allowing us to implicitly use observed labels in one task to recover unobserved labels in others. The same is true for imputing missing features. In fact, our methods learn in the difficult regime of *multi-label transductive learning with missing data* that one sometimes encounters in practice. That is, each item is associated with many class labels, many of the items' labels may be unobserved (some items may be completely unlabeled across all labels), and many features may also be unobserved. Our methods build upon recent advances in matrix completion, with efficient algorithms to handle matrices with mixed real-valued features and discrete labels. We obtain promising experimental results on a range of synthetic and real-world data.

6.1 Problem Formulation

Let $\mathbf{x}_1 \dots \mathbf{x}_n \in \mathbb{R}^d$ be feature vectors associated with n items. Let $\mathbf{X} = [\mathbf{x}_1 \dots \mathbf{x}_n]$ be the $d \times n$ feature matrix whose columns are the items. Let there be t binary classification tasks,

[1] Based on joint work with Xiaojin Zhu, Benjamin Recht, Junming Xu, and Robert Nowak.

$\mathbf{y}_1 \dots \mathbf{y}_n \in \{-1, 1\}^t$ be the label vectors, and $\mathbf{Y} = [\mathbf{y}_1 \dots \mathbf{y}_n]$ be the $t \times n$ label matrix. Entries in $\mathbf{X}$ or $\mathbf{Y}$ can be missing at random. Let $\Omega_{\mathbf{X}}$ be the index set of observed features in $\mathbf{X}$, such that $(i, j) \in \Omega_{\mathbf{X}}$ if and only if x_{ij} is observed. Similarly, let $\Omega_{\mathbf{Y}}$ be the index set of observed labels in $\mathbf{Y}$. Our main goal is to predict the missing labels y_{ij} for $(i, j) \notin \Omega_{\mathbf{Y}}$. Of course, this reduces to standard transductive learning when $t = 1$, $|\Omega_{\mathbf{X}}| = nd$ (no missing features), and $1 < |\Omega_{\mathbf{Y}}| < n$ (some missing labels). In our more general setting, as a side product we are also interested in imputing the missing features, and de-noising the observed features, in $\mathbf{X}$.

6.1.1 Model Assumptions

The above problem is in general ill-posed. We now describe our assumptions to make it a well-defined problem. In a nutshell, we assume that $\mathbf{X}$ and $\mathbf{Y}$ are jointly produced by an underlying low-rank matrix. We then take advantage of the sparsity to fill in the missing labels and features using a modified method of matrix completion. Specifically, we assume the following generative story. It starts from a $d \times n$ low-rank "pre"-feature matrix $\mathbf{X^0}$, with $\mathrm{rank}(\mathbf{X^0}) \ll \min(d, n)$. The actual feature matrix $\mathbf{X}$ is obtained by adding i.i.d. Gaussian noise to the entries of $\mathbf{X^0}$: $\mathbf{X} = \mathbf{X^0} + \boldsymbol{\epsilon}$, where $\epsilon_{ij} \sim N(0, \sigma_\epsilon^2)$. Meanwhile, the t "soft" labels $\left(y_{1j}^0 \dots y_{tj}^0\right)^\top \equiv \mathbf{y_j^0} \in \mathbb{R}^t$ of item j are produced by $\mathbf{y_j^0} = \mathbf{W}\mathbf{x_j^0} + \mathbf{b}$, where $\mathbf{W}$ is a $t \times d$ weight matrix, and $\mathbf{b} \in \mathbb{R}^t$ is a bias vector. Let $\mathbf{Y^0} = [\mathbf{y_1^0} \dots \mathbf{y_n^0}]$ be the soft label matrix. Note the combined $(t + d) \times n$ matrix $[\mathbf{Y^0}; \mathbf{X^0}]$ is low-rank, too: $\mathrm{rank}([\mathbf{Y^0}; \mathbf{X^0}]) \leq \mathrm{rank}(\mathbf{X^0}) + 1$. The actual label $y_{ij} \in \{-1, 1\}$ is generated randomly via a sigmoid function: $P(y_{ij}|y_{ij}^0) = 1/\left(1 + \exp(-y_{ij}y_{ij}^0)\right)$. Finally, two random masks $\Omega_{\mathbf{X}}, \Omega_{\mathbf{Y}}$ are applied to expose only some of the entries in $\mathbf{X}$ and $\mathbf{Y}$, and we use ω to denote the percentage of observed entries. This generative story may seem restrictive, but our approaches based on it perform well on synthetic and real datasets, outperforming several baselines with linear classifiers.

6.1.2 Matrix Completion for Heterogeneous Matrix Entries

With the above data generation model, our task can be defined as follows. Given the partially observed features and labels as specified by $\mathbf{X}, \mathbf{Y}, \Omega_{\mathbf{X}}, \Omega_{\mathbf{Y}}$, we would like to recover the intermediate low-rank matrix $[\mathbf{Y^0}; \mathbf{X^0}]$. Then, $\mathbf{X^0}$ will contain the denoised and completed features, and $\mathrm{sign}(\mathbf{Y^0})$ will contain the completed and correct labels.

The key assumption is that the $(t+d) \times n$ stacked matrix $[\mathbf{Y^0}; \mathbf{X^0}]$ is of low rank. We will start from a "hard" formulation that is illustrative but impractical, then relax it.

$$\begin{aligned} \underset{\mathbf{Z} \in \mathbb{R}^{(t+d)\times n}}{\text{argmin}} \quad & \mathrm{rank}(\mathbf{Z}) \\ \text{s.t.} \quad & \mathrm{sign}(z_{ij}) = y_{ij}, \ \ \forall (i,j) \in \Omega_{\mathbf{Y}}; \qquad z_{(i+t)j} = x_{ij}, \ \ \forall (i,j) \in \Omega_{\mathbf{X}} \end{aligned} \tag{6.1}$$

Here, $\mathbf{Z}$ is meant to recover $[\mathbf{Y^0}; \mathbf{X^0}]$ by directly minimizing the rank while obeying the observed features and labels. Note the indices $(i,j) \in \Omega_{\mathbf{X}}$ are with respect to $\mathbf{X}$, such that $i \in \{1, \ldots, d\}$. To index the corresponding element in the larger stacked matrix $\mathbf{Z}$, we need to shift the row index by t to skip the part for $\mathbf{Y^0}$, and hence the constraints $z_{(i+t)j} = x_{ij}$. The above formulation assumes that there is no noise in the generation processes $\mathbf{X^0} \to \mathbf{X}$ and $\mathbf{Y^0} \to \mathbf{Y}$. Of course, there are several issues with formulation (6.1), and we handle them as follows:

- $\mathrm{rank}()$ is a non-convex function and difficult to optimize. Following recent work in matrix completion (Candès and Tao, 2010; Candès and Recht, 2009), we relax $\mathrm{rank}()$ with the convex nuclear norm $\|\mathbf{Z}\|_* = \sum_{k=1}^{\min(t+d,n)} \sigma_k(\mathbf{Z})$, where σ_k's are the singular values of $\mathbf{Z}$. The relationship between $\mathrm{rank}(\mathbf{Z})$ and $\|\mathbf{Z}\|_*$ is analogous to that of ℓ^0-norm and ℓ^1-norm for vectors.

- There is feature noise from $\mathbf{X^0}$ to $\mathbf{X}$. Instead of the equality constraints in (6.1), we minimize a loss function $c_x(z_{(i+t)j}, x_{ij})$. We choose the squared loss $c_x(u,v) = \frac{1}{2}(u-v)^2$ in this work, but other convex loss functions are possible too.

- Similarly, there is label noise from $\mathbf{Y^0}$ to $\mathbf{Y}$. The observed labels are of a different type than the observed features. We therefore introduce another loss function $c_y(z_{ij}, y_{ij})$ to account for the heterogeneous data. In this work, we use the logistic loss $c_y(u,v) = \log(1 + \exp(-uv))$.

In addition to these changes, we will model the bias $\mathbf{b}$ either explicitly or implicitly, leading to two alternative matrix completion formulations below.

Formulation 1 (MC-b). In this formulation, we explicitly optimize the bias $\mathbf{b} \in \mathbb{R}^t$ in addition to $\mathbf{Z} \in \mathbb{R}^{(t+d)\times n}$, hence the name. Here, $\mathbf{Z}$ corresponds to the stacked matrix $[\mathbf{W}\mathbf{X}^0; \mathbf{X}^0]$ instead of $[\mathbf{Y^0}; \mathbf{X^0}]$, making it potentially lower rank. The optimization problem is

$$\underset{\mathbf{Z},\mathbf{b}}{\operatorname{argmin}} \quad \mu\|\mathbf{Z}\|_* + \frac{\lambda}{|\Omega_\mathbf{Y}|}\sum_{(i,j)\in\Omega_\mathbf{Y}} c_y(z_{ij}+b_i, y_{ij}) + \frac{1}{|\Omega_\mathbf{X}|}\sum_{(i,j)\in\Omega_\mathbf{X}} c_x(z_{(i+t)j}, x_{ij}), \tag{6.2}$$

where μ, λ are positive trade-off weights. Notice the bias $\mathbf{b}$ is not regularized. This is a convex problem, whose optimization procedure will be discussed in section 6.2. Once the optimal $\mathbf{Z}, \mathbf{b}$ are found, we recover the task-i label of item j by $\operatorname{sign}(z_{ij}+b_i)$, and feature k of item j by $z_{(k+t)j}$.

Formulation 2 (MC-1). In this formulation, the bias is modeled implicitly within $\mathbf{Z}$. Similar to how bias is commonly handled in linear classifiers, we append an additional feature with constant value one to each item. The corresponding pre-feature matrix is augmented into $\left[\mathbf{X^0}; \mathbf{1}^\top\right]$, where $\mathbf{1}$ is the all-1 vector. Under the same label assumption $\mathbf{y}_j^\mathbf{0} = \mathbf{W}\mathbf{x}_j^\mathbf{0} + \mathbf{b}$, the rows of the soft label matrix $\mathbf{Y^0}$ are linear combinations of rows in $\left[\mathbf{X^0}; \mathbf{1}^\top\right]$, i.e., $\operatorname{rank}(\left[\mathbf{Y^0}; \mathbf{X^0}; \mathbf{1}^\top\right]) = \operatorname{rank}(\left[\mathbf{X^0}; \mathbf{1}^\top\right])$. We then let $\mathbf{Z}$ correspond to the $(t+d+1)\times n$ stacked matrix $\left[\mathbf{Y^0}; \mathbf{X^0}; \mathbf{1}^\top\right]$, by forcing its last row to be $\mathbf{1}^\top$ (hence the name):

$$\begin{aligned} \underset{\mathbf{Z}\in\mathbb{R}^{(t+d+1)\times n}}{\operatorname{argmin}} \quad & \mu\|\mathbf{Z}\|_* + \frac{\lambda}{|\Omega_\mathbf{Y}|}\sum_{(i,j)\in\Omega_\mathbf{Y}} c_y(z_{ij}, y_{ij}) + \frac{1}{|\Omega_\mathbf{X}|}\sum_{(i,j)\in\Omega_\mathbf{X}} c_x(z_{(i+t)j}, x_{ij}) \\ \text{s.t.} \quad & z_{(t+d+1)\cdot} = \mathbf{1}^\top. \end{aligned} \tag{6.3}$$

This is a constrained convex optimization problem. Once the optimal $\mathbf{Z}$ is found, we recover the task-i label of item j by $\operatorname{sign}(z_{ij})$, and feature k of item j by $z_{(k+t)j}$.

MC-b and MC-1 differ mainly in what is in $\mathbf{Z}$, which leads to different behaviors of the nuclear norm. Despite the generative story, we do not explicitly recover the weight matrix $\mathbf{W}$ in these formulations. Other formulations are certainly possible. One way is to let $\mathbf{Z}$ correspond to $[\mathbf{Y^0}; \mathbf{X^0}]$ directly, without introducing bias $\mathbf{b}$ or the all-1 row, and hope nuclear norm minimization will prevail. This is inferior in our preliminary experiments, and we do not explore it further in this work.

6.2 Optimization Techniques

We solve MC-b and MC-1 using modifications of the Fixed Point Continuation (FPC) method of Ma et al. (2009).[2] While nuclear norm minimization can be converted into a semidefinite programming (SDP) problem (Candès and Recht, 2009), current SDP solvers are severely limited in the size of problems they can solve. Instead, the basic fixed point approach is a computationally efficient alternative, which provably converges to the globally optimal solution and has been shown to outperform SDP solvers in terms of matrix recoverability.

6.2.1 Fixed Point Continuation for MC-b

We first describe our modified FPC method for MC-b. It differs from the original FPC (Ma et al., 2009) in the extra bias variables and multiple loss functions. Our fixed point iterative algorithm to solve the unconstrained problem of (6.2) consists of two alternating steps for each iteration k:

1. (gradient step) $\mathbf{b}^{k+1} = \mathbf{b}^k - \tau_{\mathbf{b}} g(\mathbf{b}^k)$, $\mathbf{A}^k = \mathbf{Z}^k - \tau_{\mathbf{Z}} g(\mathbf{Z}^k)$
2. (shrinkage step) $\mathbf{Z}^{k+1} = S_{\tau_{\mathbf{Z}}\mu}(\mathbf{A}^k)$.

In the gradient step, $\tau_{\mathbf{b}}$ and $\tau_{\mathbf{Z}}$ are step sizes whose choice will be discussed next. Overloading notation a bit, $g(\mathbf{b}^k)$ is the vector gradient, and $g(\mathbf{Z}^k)$ is the matrix gradient, respectively, of the two loss terms in (6.2) (i.e., excluding the nuclear norm term):

$$g(b_i) = \frac{\lambda}{|\Omega_{\mathbf{Y}}|} \sum_{j:(i,j)\in\Omega_{\mathbf{Y}}} \frac{-y_{ij}}{1+\exp(y_{ij}(z_{ij}+b_i))} \tag{6.4}$$

$$g(z_{ij}) = \begin{cases} \frac{\lambda}{|\Omega_{\mathbf{Y}}|}\frac{-y_{ij}}{1+\exp(y_{ij}(z_{ij}+b_i))}, & i \le t \text{ and } (i,j)\in\Omega_{\mathbf{Y}} \\ \frac{1}{|\Omega_{\mathbf{X}}|}(z_{ij} - x_{(i-t)j}), & i > t \text{ and } (i-t,j)\in\Omega_{\mathbf{X}} \\ 0, & \text{otherwise} \end{cases} \tag{6.5}$$

Note for $g(z_{ij}), i > t$, we need to shift down (un-stack) the row index by t in order to map the element in $\mathbf{Z}$ back to the item $x_{(i-t)j}$.

[2] While the primary method of Ma et al. (2009) is Fixed Point Continuation with Approximate Singular Value Decomposition (FPCA), where the approximate SVD is used to speed up the algorithm, we opt to use an exact SVD for simplicity and will refer to the method simply as FPC.

In the shrinkage step, $S_{\tau_{\mathbf{Z}}\mu}(\cdot)$ is a matrix shrinkage operator. Let $\mathbf{A}^k = \mathbf{U}\Lambda\mathbf{V}^\top$ be the SVD of $\mathbf{A}^k$. Then $S_{\tau_{\mathbf{Z}}\mu}(\mathbf{A}^k) = \mathbf{U}\max(\Lambda - \tau_{\mathbf{Z}}\mu, 0)\mathbf{V}^\top$, where $\max$ is elementwise. That is, the shrinkage operator shifts the singular values down, and truncates any negative values to zero. This step reduces the nuclear norm.

Even though the problem is convex, convergence can be slow. We follow Ma et al. (2009) and use a continuation or homotopy method to improve the speed. This involves beginning with a large value $\mu_1 > \mu$ and solving a sequence of subproblems, each with a decreasing value and using the previous solution as its initial point. The sequence of values is determined by a decay parameter η_μ: $\mu_{k+1} = \max\{\mu_k\eta_\mu, \mu\}, \quad k = 1, \ldots, L-1$, where μ is the final value to use, and L is the number of rounds of continuation. The complete FPC algorithm for MC-b is listed in Algorithm 7.

A minor modification of the argument in Ma et al. (2009) reveals that as long as we choose non-negative step sizes satisfying $\tau_{\mathbf{b}} < 4|\Omega_{\mathbf{Y}}|/(\lambda n)$ and $\tau_{\mathbf{Z}} < \min\{4|\Omega_{\mathbf{Y}}|/\lambda, |\Omega_{\mathbf{X}}|\}$, the algorithms MC-b will be guaranteed to converge to a global optimum. Indeed, to guarantee convergence, we only need that the gradient step is *non-expansive* in the sense that

$$\|\mathbf{b}_1 - \tau_{\mathbf{b}}g(\mathbf{b}_1) - \mathbf{b}_2 + \tau_{\mathbf{b}}g(\mathbf{b}_2)\|^2 + \|\mathbf{Z}_1 - \tau_{\mathbf{Z}}g(\mathbf{Z}_1) - \mathbf{Z}_2 + \tau_{\mathbf{Z}}g(\mathbf{Z}_2)\|_F^2 \leq \|\mathbf{b}_1 - \mathbf{b}_2\|^2 + \|\mathbf{Z}_1 - \mathbf{Z}_2\|_F^2$$

for all $\mathbf{b}_1$, $\mathbf{b}_2$, $\mathbf{Z}_1$, and $\mathbf{Z}_2$. Our choice of $\tau_{\mathbf{b}}$ and $\tau_{\mathbf{Z}}$ guarantee such non-expansiveness. Once this non-expansiveness is satisfied, the remainder of the convergence analysis is the same as in Ma et al. (2009).

6.2.2 Fixed Point Continuation for MC-1

Our modified FPC method for MC-1 is similar except for two differences. First, there is no bias variable $\mathbf{b}$. Second, the shrinkage step will in general not satisfy the all-1-row constraints in (6.3). Thus, we add a third projection step at the end of each iteration to project $\mathbf{Z}^{k+1}$ back to the feasible region, by simply setting its last row to all 1's. The complete algorithm for MC-1 is given in Algorithm 8. We were unable to prove convergence for this gradient + shrinkage + projection algorithm. Nonetheless, in our empirical experiments, Algorithm 8 always converges and tends to outperform MC-b. The two algorithms have about the same convergence speed.

Input: Initial matrix $\mathbf{Z}_0$, bias $\mathbf{b}_0$,
parameters μ, λ, Step sizes $\tau_{\mathbf{b}}, \tau_{\mathbf{Z}}$
Determine $\mu_1 > \mu_2 > \cdots > \mu_L = \mu > 0$.
Set $\mathbf{Z} = \mathbf{Z}_0, \mathbf{b} = \mathbf{b}_0$.
foreach $\mu = \mu_1, \mu_2, \ldots, \mu_L$ **do**
  while *Not converged* **do**
    Compute $\mathbf{b} = \mathbf{b} - \tau_{\mathbf{b}} g(\mathbf{b})$,
    $\mathbf{A} = \mathbf{Z} - \tau_{\mathbf{Z}} g(\mathbf{Z})$
    Compute SVD of $\mathbf{A} = \mathbf{U}\Lambda\mathbf{V}^\top$
    Compute
    $\mathbf{Z} = \mathbf{U}\max(\Lambda - \tau_{\mathbf{Z}}\mu, 0)\mathbf{V}^\top$
  end
end
Output: Recovered matrix $\mathbf{Z}$, bias $\mathbf{b}$

Algorithm 7: FPC algorithm for MC-b.

Input: Initial matrix $\mathbf{Z}_0$,
parameters μ, λ, Step size $\tau_{\mathbf{Z}}$
Determine $\mu_1 > \mu_2 > \cdots > \mu_L = \mu > 0$.
Set $\mathbf{Z} = \mathbf{Z}_0$.
foreach $\mu = \mu_1, \mu_2, \ldots, \mu_L$ **do**
  while *Not converged* **do**
    Compute $\mathbf{A} = \mathbf{Z} - \tau_{\mathbf{Z}} g(\mathbf{Z})$
    Compute SVD of $\mathbf{A} = \mathbf{U}\Lambda\mathbf{V}^\top$
    Compute
    $\mathbf{Z} = \mathbf{U}\max(\Lambda - \tau_{\mathbf{Z}}\mu, 0)\mathbf{V}^\top$
    Project $\mathbf{Z}$ to feasible region
    $z_{(t+d+1)\cdot} = \mathbf{1}^\top$
  end
end
Output: Recovered matrix $\mathbf{Z}$

Algorithm 8: FPC algorithm for MC-1.

6.3 Experiments

We now empirically study the ability of matrix completion to perform multi-label transductive classification when there is missing data. We first present a family of 24 experiments on a synthetic task by systematically varying different aspects of the task, including the rank of the problem, noise level, number of items, and observed label and feature percentage. We then present experiments on two real-world datasets: music emotions and yeast microarray. In each experiments, we compare MC-b and MC-1 against four other baseline algorithms. Our results show that MC-1 consistently outperforms other methods, and MC-b follows closely.

Parameter Tuning and Other Settings for MC-b and MC-1: To tune the parameters μ and λ, we use 5-fold cross validation (CV) separately for each experiment. Specifically, we randomly divide $\Omega_{\mathbf{X}}$ and $\Omega_{\mathbf{Y}}$ into five disjoint subsets each. We then run our matrix completion algorithms using $\frac{4}{5}$ of the observed entries, measure its performance on the remaining $\frac{1}{5}$, and average over the five folds. Since our main goal is to predict unobserved labels, we use label error as the CV performance criterion to select parameters. Note that tuning μ is quite efficient since all values under consideration can be evaluated in one run of the continuation method. We set $\eta_\mu = 0.25$

and, as in Ma et al. (2009), consider μ values starting at $\sigma_1\eta_\mu$, where σ_1 is the largest singular value of the matrix of observed entries in $[\mathbf{Y};\mathbf{X}]$ (with the unobserved entries set to 0), and decrease μ until 10^{-5}. The range of λ values considered was $\{10^{-3}, 10^{-2}, 10^{-1}, 1\}$. We initialized $\mathbf{b}_0$ to be all zero and $\mathbf{Z}_0$ to be the rank-1 approximation of the matrix of observed entries in $[\mathbf{Y};\mathbf{X}]$ (with unobserved entries set to 0) obtained by performing an SVD and reconstructing the matrix using only the largest singular value and corresponding left and right singular vectors. The step sizes were set as follows: $\tau_{\mathbf{Z}} = \min(\frac{3.8|\Omega_{\mathbf{Y}}|}{\lambda}, |\Omega_{\mathbf{X}}|)$, $\tau_{\mathbf{b}} = \frac{3.8|\Omega_{\mathbf{Y}}|}{\lambda n}$. Convergence was defined as relative change in objective functions (6.2)(6.3) smaller than 10^{-5}.

Baselines: We compare to the following baselines, each consisting of some missing feature imputation step on $\mathbf{X}$ first, then using a standard SVM to predict the labels:

- **[FPC+SVM]** Matrix completion on $\mathbf{X}$ alone using FPC (Ma et al., 2009).
- **[EM(k)+SVM]** Expectation Maximization algorithm to impute missing $\mathbf{X}$ entries using a mixture of k Gaussian components. Missing features, mixing component parameters, and the assignments of items to components are treated as hidden variables, which are estimated in an iterative manner to maximize the likelihood of the data (Little and Rubin, 2002).
- **[Mean+SVM]** Impute each missing feature by the mean of the observed entries for that feature.
- **[Zero+SVM]** Impute missing features by filling in zeros.

After imputation, an SVM is trained using the available (noisy) labels in $\Omega_{\mathbf{Y}}$ for that task, and predictions are made for the rest of the labels. All SVMs are linear, trained using SVMlin[3], and the regularization parameter is tuned using 5-fold cross validation separately for each task. The range of parameter values considered was $\{10^{-8}, 10^{-7}, \ldots, 10^{7}, 10^{8}\}$.

Evaluation Method: To evaluate performance, we consider two measures: *transductive label error*, i.e., the percentage of unobserved labels predicted incorrectly; and *relative feature imputation error* $\left(\sum_{ij\notin\Omega_{\mathbf{X}}}(x_{ij}-\hat{x}_{ij})^2\right)/\sum_{ij\notin\Omega_{\mathbf{X}}} x_{ij}^2$, where $\hat{x}$ is the predicted feature value. In the tables

[3] http://vikas.sindhwani.org/svmlin.html

below, for each parameter setting, we report the mean performance (and standard deviation in parenthesis) of different algorithms over 10 random trials. The best algorithm within each parameter setting, as well as any statistically indistinguishable algorithms via a two-tailed paired t-test at significance level $\alpha = 0.05$, are marked in bold.

6.3.1 Synthetic Data Experiments

Synthetic Data Generation: We generate a family of synthetic datasets to systematically explore the performance of the algorithms. We first create a rank-r matrix $\mathbf{X^0} = \mathbf{L}\mathbf{R}^\top$, where $\mathbf{L} \in \mathbb{R}^{d\times r}$ and $\mathbf{R} \in \mathbb{R}^{n\times r}$ with entries drawn i.i.d. from $\mathcal{N}(0, 1)$. We then normalize $\mathbf{X^0}$ such that its entries have variance 1. Next, we create a weight matrix $\mathbf{W} \in \mathbb{R}^{t\times d}$ and bias vector $\mathbf{b} \in \mathbb{R}^t$, with all entries drawn i.i.d. from $\mathcal{N}(0, 10)$. We then produce $\mathbf{X}, \mathbf{Y^0}, \mathbf{Y}$ according to section 6.1.1. Finally, we produce the random $\Omega_\mathbf{X}, \Omega_\mathbf{Y}$ masks with ω percent observed entries.

Using the above procedure, we vary $\omega = 10\%, 20\%, 40\%$, $n = 100, 400$, $r = 2, 4$, and $\sigma_\epsilon^2 = 0.01, 0.1$, while fixing $t = 10, d = 20$, to produce 24 different parameter settings. For each setting, we generate 10 trials, where the randomness is in the data and mask.

Synthetic experiment results: Table 6.1 shows the transductive label errors, and Table 6.2 shows the relative feature imputation errors, on the synthetic datasets. We make several observations.

Observation 1: MC-b and MC-1 are the best for feature imputation, as Table 6.2 shows. However, the imputations are not perfect, because in these particular parameter settings the ratio between the number of observed entries over the degrees of freedom needed to describe the feature matrix (i.e., $r(d + n - r)$) is below the necessary condition for perfect matrix completion (Candès and Recht, 2009), and because there is some feature noise. Furthermore, our CV tuning procedure selects parameters μ, λ to optimize label error, which often leads to suboptimal imputation performance. In a separate experiment (not reported here) when we made the ratio sufficiently large and without noise, and specifically tuned for imputation error, both MC-b and MC-1 did achieve perfect feature imputation. Also, FPC+SVM is slightly worse in feature imputation. This may

σ_ϵ^2	r	n	ω	MC-b	MC-1	FPC+SVM	EM1+SVM	Mean+SVM	Zero+SVM
0.01	2	100	10%	37.8(4.0)	**31.8(4.3)**	**34.8(7.0)**	34.6(3.9)	40.5(5.7)	40.5(5.1)
			20%	23.5(2.9)	**17.0(2.2)**	**17.6(2.1)**	19.7(2.4)	28.7(4.1)	27.4(4.4)
			40%	15.1(3.1)	10.8(1.8)	**9.6(1.5)**	10.4(1.0)	16.5(2.5)	15.4(2.3)
		400	10%	26.5(2.0)	**19.9(1.7)**	23.7(1.7)	24.2(1.9)	32.4(2.9)	31.5(2.7)
			20%	15.9(2.5)	**11.7(1.9)**	12.6(2.2)	**12.0(1.9)**	20.0(1.9)	19.7(1.7)
			40%	11.7(2.0)	8.0(1.6)	**7.2(1.8)**	**7.3(1.4)**	12.2(1.8)	12.1(2.0)
	4	100	10%	42.5(4.0)	**40.8(4.4)**	**41.5(2.6)**	**43.2(2.2)**	43.5(2.9)	**42.9(2.9)**
			20%	33.2(2.3)	**26.2(2.8)**	**26.7(1.7)**	30.8(2.7)	35.5(1.4)	33.9(1.5)
			40%	19.6(3.1)	**14.3(2.7)**	**13.6(2.6)**	**14.1(2.4)**	22.5(2.0)	21.7(2.3)
		400	10%	35.3(3.1)	**32.1(1.6)**	**33.4(1.6)**	34.2(1.8)	37.7(1.2)	38.2(1.4)
			20%	24.4(2.3)	**19.1(1.3)**	20.5(1.4)	**19.8(1.1)**	26.9(1.5)	26.9(1.3)
			40%	14.6(1.8)	9.5(0.5)	9.2(0.9)	**8.6(1.1)**	16.4(1.2)	16.5(1.3)
0.1	2	100	10%	39.6(5.5)	**34.6(3.5)**	**37.3(6.4)**	40.2(5.3)	41.5(6.0)	41.0(5.7)
			20%	25.2(2.6)	**20.1(1.7)**	**21.6(2.6)**	26.8(3.7)	31.8(4.7)	29.9(4.0)
			40%	15.7(3.1)	**12.6(1.4)**	**13.2(2.0)**	15.1(2.4)	18.5(2.7)	17.2(2.4)
		400	10%	27.6(2.1)	**22.6(1.9)**	27.6(2.4)	28.8(2.6)	34.5(3.3)	33.6(2.8)
			20%	18.0(2.2)	**15.2(1.7)**	16.8(2.3)	18.4(2.5)	22.6(2.4)	21.8(2.5)
			40%	12.0(2.1)	**10.1(1.3)**	**10.4(2.1)**	11.1(1.9)	14.1(2.0)	14.0(2.4)
	4	100	10%	**42.5(4.3)**	**41.5(2.5)**	**42.3(2.0)**	45.6(1.9)	44.6(2.9)	43.6(2.3)
			20%	33.3(1.9)	**29.0(2.2)**	**30.9(3.1)**	34.9(3.0)	36.2(2.3)	35.4(1.6)
			40%	21.4(2.7)	**18.4(3.1)**	**18.7(2.4)**	21.6(2.4)	23.9(2.0)	23.3(2.5)
		400	10%	36.3(2.7)	**34.0(1.7)**	**35.1(1.2)**	36.3(1.4)	38.7(1.3)	39.1(1.2)
			20%	25.5(2.0)	**21.8(1.0)**	23.8(1.5)	25.1(1.4)	28.4(1.7)	28.4(1.8)
			40%	16.0(1.8)	**12.8(0.8)**	13.9(1.2)	14.7(1.3)	18.3(1.2)	18.2(1.2)
meta-average				25.6	21.4	22.6	24.1	28.6	28.0

Table 6.1: Transductive label error of six algorithms on the 24 synthetic datasets. The varying parameters are feature noise σ_ϵ^2, $\mathrm{rank}(\mathbf{X^0}) = r$, number of items n, and observed label and feature percentage ω. Each row is for a unique parameter combination. Each cell shows the mean(standard deviation) of transductive label error (in percentage) over 10 random trials. The "meta-average" row is the simple average over all parameter settings and all trials. The best algorithm within each parameter setting (row), as well as any statistically indistinguishable algorithms via a two-tailed, paired t-test at significance level $\alpha = 0.05$, are marked in bold.

seem curious as FPC focuses exclusively on imputing $\mathbf{X}$. We believe the fact that MC-b and MC-1 can use information in $\mathbf{Y}$ to enhance feature imputation in $\mathbf{X}$ made them better than FPC+SVM.

σ_ϵ^2	r	n	ω	MC-b	MC-1	FPC+SVM	EM1+SVM	Mean+SVM
0.01	2	100	10%	**0.84(0.04)**	**0.87(0.06)**	**0.88(0.06)**	1.01(0.12)	1.06(0.02)
			20%	**0.54(0.08)**	**0.57(0.06)**	**0.57(0.07)**	0.67(0.13)	1.03(0.02)
			40%	**0.29(0.06)**	**0.27(0.06)**	**0.27(0.06)**	0.34(0.03)	1.01(0.01)
		400	10%	**0.73(0.03)**	**0.72(0.04)**	0.76(0.03)	0.79(0.07)	1.02(0.01)
			20%	**0.43(0.04)**	0.46(0.05)	0.50(0.04)	**0.45(0.04)**	1.01(0.00)
			40%	0.30(0.10)	**0.22(0.04)**	0.24(0.05)	**0.21(0.04)**	1.00(0.00)
	4	100	10%	**0.99(0.04)**	**0.96(0.03)**	**0.96(0.03)**	1.22(0.11)	1.05(0.01)
			20%	**0.77(0.05)**	**0.78(0.05)**	**0.77(0.04)**	0.92(0.07)	1.02(0.01)
			40%	**0.42(0.07)**	**0.40(0.03)**	0.42(0.04)	0.49(0.04)	1.01(0.01)
		400	10%	**0.87(0.04)**	**0.88(0.03)**	0.89(0.01)	1.00(0.08)	1.01(0.00)
			20%	**0.69(0.07)**	**0.67(0.04)**	0.69(0.03)	**0.66(0.03)**	1.01(0.00)
			40%	0.34(0.05)	0.34(0.03)	0.38(0.03)	**0.29(0.02)**	1.00(0.00)
0.1	2	100	10%	**0.92(0.05)**	**0.93(0.04)**	**0.93(0.05)**	1.18(0.10)	1.06(0.02)
			20%	**0.69(0.07)**	0.72(0.06)	0.74(0.06)	0.94(0.07)	1.03(0.02)
			40%	**0.51(0.05)**	**0.52(0.05)**	0.53(0.05)	0.67(0.08)	1.02(0.01)
		400	10%	**0.79(0.03)**	0.80(0.03)	0.84(0.03)	0.96(0.07)	1.02(0.01)
			20%	**0.64(0.06)**	**0.64(0.06)**	0.67(0.04)	0.73(0.07)	1.01(0.00)
			40%	0.48(0.04)	**0.45(0.05)**	0.49(0.05)	0.57(0.07)	1.00(0.00)
	4	100	10%	**1.01(0.04)**	**0.97(0.03)**	**0.97(0.03)**	1.25(0.05)	1.05(0.02)
			20%	**0.84(0.03)**	**0.85(0.03)**	**0.85(0.03)**	1.07(0.06)	1.02(0.01)
			40%	**0.59(0.03)**	**0.61(0.04)**	0.63(0.04)	0.80(0.09)	1.01(0.01)
		400	10%	**0.90(0.02)**	**0.92(0.02)**	0.92(0.01)	1.08(0.07)	1.01(0.01)
			20%	**0.75(0.04)**	**0.77(0.02)**	0.79(0.03)	0.86(0.05)	1.01(0.00)
			40%	0.56(0.03)	**0.55(0.04)**	0.59(0.04)	0.66(0.06)	1.00(0.00)
meta-average				0.66	0.66	0.68	0.78	1.02

Table 6.2: Relative feature imputation error on the synthetic datasets. The algorithm Zero+SVM is not shown because it by definition has relative feature imputation error 1.

Observation 2: MC-1 is the best for multi-label transductive classification, as suggested by Table 6.1. Surprisingly, the feature imputation advantage of MC-b did not translate into classification, and FPC+SVM took second place.

Observation 3: The same factors that affect standard matrix completion also affect classification performance of MC-b and MC-1. As the tables show, everything else being equal, less feature noise (smaller σ_ϵ^2), lower rank r, more items, or more observed features and labels, reduce label error. Beneficial combination of these factors (the 6^{th} row) produces the lowest label errors.

t	MC-b	MC-1	FPC+SVM	MC-b	MC-1	FPC+SVM
2	30.1(2.8)	22.9(2.2)	20.5(2.5)	0.78(0.07)	0.78(0.04)	0.76(0.03)
10	**26.5(2.0)**	**19.9(1.7)**	23.7(1.7)	**0.73(0.03)**	**0.72(0.04)**	0.76(0.03)
	transductive label error			relative feature imputation error		

Table 6.3: More tasks help matrix completion ($\omega = 10\%$, $n = 400$, $r = 2$, $d = 20$, $\sigma_\epsilon^2 = 0.01$).

Matrix completion benefits from more tasks. We performed one additional synthetic data experiment examining the effect of t (the number of tasks) on MC-b and MC-1, with the remaining data parameters fixed at $\omega = 10\%$, $n = 400$, $r = 2$, $d = 20$, and $\sigma_\epsilon^2 = 0.01$. Table 6.3 reveals that both MC methods achieve statistically significantly better label prediction and imputation performance with $t = 10$ than with only $t = 2$ (as determined by two-sample t-tests at significance level 0.05).

6.3.2 Music Emotions Data Experiments

In this task introduced by Trohidis et al. (2008), the goal is to predict which of several types of emotion are present in a piece of music. The data[4] consists of $n = 593$ songs of a variety of musical genres, each labeled with one or more of $t = 6$ emotions (i.e., amazed-surprised, happy-pleased, relaxing-calm, quiet-still, sad-lonely, and angry-fearful). Each song is represented by $d = 72$ features (8 rhythmic, 64 timbre-based) automatically extracted from a 30-second sound clip.

We vary the percentage of observed entries $\omega = 40\%, 60\%, 80\%$. For each ω, we run 10 random trials with different masks $\Omega_{\mathbf{X}}, \Omega_{\mathbf{Y}}$. For this dataset, we tuned only μ with CV, and set $\lambda = 1$.

The results are in Table 6.4. Most importantly, these results show that MC-1 is useful for this real-world multi-label classification problem, leading to the best (or statistically indistinguishable from the best) transductive error performance with 60% and 80% of the data available, and close to the best with only 40%.

[4] Available at http://mulan.sourceforge.net/datasets.html

ω =40%	60%	80%
28.0(1.2)	25.2(1.0)	22.2(1.6)
27.4(0.8)	**23.7(1.6)**	**19.8(2.4)**
26.9(0.7)	25.2(1.6)	24.4(2.0)
26.0(1.1)	**23.6(1.1)**	21.2(2.3)
26.2(0.9)	**23.1(1.2)**	21.6(1.6)
26.3(0.8)	24.2(1.0)	22.6(1.3)
30.3(0.6)	28.9(1.1)	25.7(1.4)

transductive label error

Algorithm
MC-b
MC-1
FPC+SVM
EM1+SVM
EM4+SVM
Mean+SVM
Zero+SVM

ω =40%	60%	80%
0.69(0.05)	0.54(0.10)	0.41(0.02)
0.60(0.05)	0.46(0.12)	0.25(0.03)
0.64(0.01)	0.46(0.02)	0.31(0.03)
0.46(0.09)	0.23(0.04)	**0.13(0.01)**
0.49(0.10)	0.27(0.04)	0.15(0.02)
0.18(0.00)	**0.19(0.00)**	0.20(0.01)
1.00(0.00)	1.00(0.00)	1.00(0.00)

relative feature imputation error

Table 6.4: Performance on the music emotions data.

We also compared these algorithms against an "oracle baseline" (not shown in the table). In this baseline, we give 100% features (i.e., no indices are missing from $\Omega_{\mathbf{X}}$) and the training labels in $\Omega_{\mathbf{Y}}$ to a standard SVM, and let it predict the unspecified labels. On the same random trials, for observed percentage $\omega = 40\%, 60\%, 80\%$, the oracle baseline achieved label error rate $22.1(0.8), 21.3(0.8), 20.5(1.8)$ respectively. Interestingly, MC-1 with $\omega = 80\%$ (19.8) is statistically indistinguishable from the oracle baseline.

6.3.3 Yeast Microarray Data Experiments

This dataset comes from a biological domain and involves the problem of Yeast gene functional classification with data originally from Eisen et al. (1998). This dataset was previously studied in the context of multi-label prediction by Elisseeff and Weston (2001).[5] The dataset contains $n = 2417$ examples (Yeast genes) with $d = 103$ input features (results from microarray experiments). Each gene belongs to one or more of 190 functional classes that form a tree-structured hierarchy. As in Elisseeff and Weston (2001), we focus on predicting each gene's membership in the $t = 14$ functional classes in the first level of the hierarchy. For this larger dataset, we omitted the computationally expensive EM4+SVM methods, and tuned only μ for matrix completion while fixing $\lambda = 1$.

Table 6.5 reveals that MC-b leads to statistically significantly lower transductive label error for this biological dataset. Although not highlighted in the table, MC-1 is also statistically better than

[5] Available at http://mulan.sourceforge.net/datasets.html

ω =40%	60%	80%
16.1(0.3)	**12.2(0.3)**	**8.7(0.4)**
16.7(0.3)	13.0(0.2)	**8.5(0.4)**
21.5(0.3)	20.8(0.3)	20.3(0.3)
22.0(0.2)	21.2(0.2)	20.4(0.2)
21.7(0.2)	21.1(0.2)	20.5(0.4)
21.6(0.2)	21.1(0.2)	20.5(0.4)

transductive label error

Algorithm
MC-b
MC-1
FPC+SVM
EM1+SVM
Mean+SVM
Zero+SVM

ω =40%	60%	80%
0.83(0.02)	0.76(0.00)	0.73(0.02)
0.86(0.00)	0.92(0.00)	0.74(0.00)
0.81(0.00)	**0.76(0.00)**	**0.72(0.00)**
1.15(0.02)	1.04(0.02)	0.77(0.01)
1.00(0.00)	1.00(0.00)	1.00(0.00)
1.00(0.00)	1.00(0.00)	1.00(0.00)

relative feature imputation error

Table 6.5: Performance on the yeast data.

the SVM methods in label error. In terms of feature imputation performance, the MC methods are weaker than FPC+SVM. However, it seems simultaneously predicting the missing labels and features appears to provide a large advantage to the MC methods. It should be pointed out that all algorithms except Zero+SVM in fact have small but non-zero standard deviation on imputation error, despite what the fixed-point formatting in the table suggests. For instance, with $\omega = 40\%$, the standard deviation is 0.0009 for MC-1, 0.0011 for FPC+SVM, and 0.0001 for Mean+SVM.

Again, we compared these algorithms to an oracle SVM baseline with 100% observed entries in $\Omega_{\mathbf{X}}$. The oracle SVM approach achieves label error of 20.9(0.1), 20.4(0.2), and 20.1(0.3) for ω =40%, 60%, and 80% observed labels, respectively. Both MC-b and MC-1 significantly outperform this oracle under paired t-tests at significance level 0.05. We attribute this advantage to a combination of multi-label learning and transduction that is intrinsic to our matrix completion methods.

6.4 Discussions and Future Work

We have introduced two matrix completion methods for multi-label transductive learning with missing features, which outperformed several baselines. In terms of problem formulation, our methods differ considerably from sparse multi-task learning (Obozinski et al., 2010; Argyriou et al., 2010; Srebro and Shraibman, 2005) in that we regularize the feature and label matrix directly, without ever learning explicit weight vectors. Our methods also differ from multi-label prediction via reduction to binary classification or ranking (Tsoumakas et al., 2010), and via compressed

sensing (Hsu et al., 2009), which assumes sparsity in that each item has a small number of positive labels, rather than the low-rank nature of feature matrices. These methods do not naturally allow for missing features. Yet other multi-label methods identify a subspace of highly predictive features across tasks in a first stage, and learn in this subspace in a second stage (Ji et al., 2008; Rai and Daume, 2009). Our methods do not require separate stages. Learning in the presence of missing data typically involves imputation followed by learning with completed data (Little and Rubin, 2002). Our methods perform imputation plus learning in one step, similar to EM on missing labels and features (Ghahramani and Jordan, 1994), but the underlying model assumption is quite different.

A drawback of our methods is their restriction to linear classifiers only. One future extension is to explicitly map the columns of the partial feature matrix to a higher dimensional space via a polynomial (or other) kernel, and apply our methods there (using a new feature matrix with additional rows). Though such mapping proliferates the missing entries, we hope that the low-rank structure in the kernelized matrix will allow us to recover labels that are nonlinear functions of the original features.

Chapter 7

Dissimilarity in Semi-Supervised Learning

A common theme in this work is the development of new semi-supervised regularizers Ω_{SSL} that can be plugged into a general risk minimization framework (see Section 1.1). As discussed earlier, such a regularizer encodes assumptions related to unlabeled data. While some well known regularizers like that of S3VMs are domain independent, Ω_{SSL} can also encode domain knowledge or other forms of weak supervision.

In this chapter, we introduce a semi-supervised classification algorithm that learns from both dissimilarity and similarity information on labeled and unlabeled data (Goldberg et al., 2007). That is, we focus on encoding a particular form of domain knowledge in Ω_{SSL}: label dissimilarity between examples, which specifies that two examples probably have different class labels. We assume we are given a set of dissimilarity pairs $\mathcal{D} = \{(i, j)\}$. For $(i, j) \in \mathcal{D}$, the two points $\mathbf{x}_i, \mathbf{x}_j$ may be both unlabeled, or one labeled and the other unlabeled. In either case we know they probably do not belong to the same class. The dissimilarity knowledge can be noisy, however. Our approach uses a novel graph-based encoding of dissimilarity that results in a convex problem, and can handle both binary and multiclass classification.

As an example, consider the problem of predicting a person's political view (left, right) from his/her postings to online blogs. The fact that person B quotes person A and uses expletives near the quote is a strong indication that B disagrees with A (Mullen and Malouf, 2006). Simple text processing thus allows us to create a dissimilarity pair (A,B) to reflect our knowledge that A and B probably have different political views.

Such dissimilarity knowledge has been extensively studied in semi-supervised clustering, introduced in Section 1.2 (Basu et al., 2006; Grira et al., 2004; Van Gael and Zhu, 2007; Wagstaff et al.,

2001; Xing et al., 2002). Recall that in this setting, pairs are known as "cannot-links" meaning they cannot be in the same cluster. These methods either directly modify the clustering algorithm, or change the underlying distance metric. Our method is different in that it specifically applies to classification, and works on discriminant functions. Dissimilarity as negative correlation on discriminant functions has been discussed in relational learning with Gaussian processes (Chu et al., 2006), but their formulation is non-convex and applies only to binary classification.

Our contribution is a convex method that incorporates both similarity and dissimilarity in semi-supervised learning. Existing graph-based semi-supervised learning methods encode *label similarity* knowledge, but they cannot handle dissimilarity easily, as we describe in Section 7.1. We define a mixed graph to accommodate both, and define the analog of the graph Laplacian. We then adapt manifold regularization (Belkin et al., 2006; Sindhwani et al., 2005a) to the mixed graph. We extend our method to multiclass classification in Section 7.2, and present experimental results in Section 7.3.

7.1 Dissimilarity in Binary Classification

In this work, we use the familiar setting where there are n items, of which l are labeled: $\{(\mathbf{x}_1, y_1), \cdots, (\mathbf{x}_l, y_l), \mathbf{x}_{l+1}, \cdots, \mathbf{x}_n\}$. Existing graph-based methods cannot easily handle dissimilarity, which is the requirement that two items have different labels. A small or zero weight w_{ij} does *not* represent dissimilarity between $\mathbf{x}_i$ and $\mathbf{x}_j$; in fact, a zero edge weight means no preference at all. A negative weight $w_{ij} < 0$ does encourage a large difference between $f(\mathbf{x}_i), f(\mathbf{x}_j)$, but this creates a number of problems. First f needs to be bounded or $\{-\infty, \infty\}$ will be a trivial minimizer. Second, any negative weight in W will make the graph energy (2.13), and ultimately the whole semi-supervised problem, non-convex. One has to resort to approximations (Ravikumar and Lafferty, 2006; Wainwright et al., 2005; Weiss and Freeman, 2001). It is highly desirable to keep the optimization problem convex.

Let us assume $y \in \{-1, 1\}$ for binary classification. Our key idea is to encode dissimilarity between i, j as $w_{ij}(f(\mathbf{x}_i) + f(\mathbf{x}_j))^2$. Note the summation. This term is zero if $f(\mathbf{x}_i), f(\mathbf{x}_j)$ have the same absolute value but opposite signs, thus encouraging different labels. The trivial case

$f(\mathbf{x}_i) = f(\mathbf{x}_j) = 0$ is avoided by competing terms in a risk minimization framework (see below). The weight w_{ij} remains positive and represents the strength of our belief in this dissimilarity edge.

A mixed graph over n nodes has similarity and dissimilarity edges, and is represented by two $n \times n$ matrices S and W. S specifies the edge type: $s_{ij} = 1$ if there is a similarity edge between i, j; $s_{ij} = -1$ if there is a dissimilarity edge. Non-negative weights $w_{ij} \geq 0$ represent the strength of the edge, regardless of its type.

The graphs in existing graph-based semi-supervised learning methods can be viewed as having an all-one S and the same W. Extending (2.13) to the mixed graph, we would like to minimize a new penalty term

$$\frac{1}{2} \sum_{i,j=1}^{n} w_{ij}(f(\mathbf{x}_i) - s_{ij} f(\mathbf{x}_j))^2. \tag{7.1}$$

It handles both similarity and dissimilarity, and is clearly convex in f. Furthermore, we can rewrite (7.1) in a quadratic form.

Let $\mathcal{M} = \mathcal{L} + (\mathbf{1} - S) \bullet W$, where $\mathcal{L}$ is the combinatorial graph Laplacian, $\mathbf{1}$ is the all-one matrix, and $\bullet$ is the Hadamard (elementwise) product. Then $\mathcal{M}$ is positive semi-definite, and

$$\mathbf{f}^\top \mathcal{M} \mathbf{f} = \frac{1}{2} \sum_{i,j} w_{ij}(f(\mathbf{x}_i) - s_{ij} f(\mathbf{x}_j))^2.$$

Therefore, the matrix $\mathcal{M}$ is the mixed-graph analog of the (unnormalized) graph Laplacian $\mathcal{L}$. Note that if the graph has no dissimilarity edges, then $\mathcal{M} = \mathcal{L}$.

We now show how to use this mixed graph to incorporate dissimilarity in the context of manifold regularization (Belkin et al., 2006). Recall that manifold regularization generalizes graph-based semi-supervised learning with a regularized risk minimization framework. Let $\mathcal{H}$ be the Reproducing Kernel Hilbert Space (RKHS) of a kernel K. Manifold regularization obtains the discriminant function by solving

$$\min_{f \in \mathcal{H}} \sum_{i=1}^{l} c(y_i, f(\mathbf{x}_i)) + \lambda_1 \|f\|_{\mathcal{H}}^2 + \lambda_2 \mathbf{f}^\top \mathcal{L} \mathbf{f}, \tag{7.2}$$

where $c()$ is an arbitrary loss function. As before, $\mathbf{f}$ is the vector of discriminant function values on the n points. The first two terms in (7.2) are the same as in supervised learning, while the third

term is the additional regularization term for graph-based semi-supervised learning. Because f is defined in $\mathcal{H}$ now, it naturally extends to new test points. Noisy labels are tolerated by the loss function.

The mixed-graph analog of (7.2) is

$$\min_{f\in\mathcal{H}} \sum_{i=1}^{l} c(y_i, f(\mathbf{x}_i)) + \lambda_1 \|f\|_{\mathcal{H}}^2 + \lambda_2 \mathbf{f}^\top \mathcal{M}\mathbf{f}. \tag{7.3}$$

One can solve the optimization problem (7.3) directly. Alternatively one can view the second and third terms together as regularization by a warped kernel, as proposed by Sindhwani et al. (2005a). In this view, one defines a second RKHS $\mathcal{H}'$, which has the same functions as $\mathcal{H}$ but a different inner product: $\langle f, g\rangle_{\mathcal{H}'} = \langle f, g\rangle_{\mathcal{H}} + f^\top M g$, where M is some positive semi-definite matrix on the n points. It follows that $\|f\|_{\mathcal{H}'}^2 = \|f\|_{\mathcal{H}}^2 + \mathbf{f}^\top M\mathbf{f}$. The *supervised* problem

$$\min_{f\in\mathcal{H}'} \sum_{i=1}^{l} c(y_i, f(\mathbf{x}_i)) + \lambda_1 \|f\|_{\mathcal{H}'}^2$$

is then equivalent to our semi-supervised learning problem (7.3), if we let $M = \frac{\lambda_2}{\lambda_1}\mathcal{M}$. Importantly, it is shown in Sindhwani et al. (2005a) that the kernel K' for the warped RKHS $\mathcal{H}'$ is related to the original K as follows:

$$k'(x, z) = k(x, z) - \mathbf{k}_x{}^\top (I + MK)^{-1} M \mathbf{k}_z, \tag{7.4}$$

where $\mathbf{k}_x = (k(\mathbf{x}_1, \mathbf{x}), \cdots, k(\mathbf{x}_n, \mathbf{x}))^\top$. This allows one to compute the warped kernel K' from some original kernel (e.g., RBF) K and the mixed-graph $\mathcal{M}$. Therefore, to solve (7.3), we can use K' in conjunction with standard supervised kernel machine software.

7.2 Dissimilarity in Multiclass Classification

It is non-trivial to incorporate dissimilarity into multiclass classification.

1. One-vs-rest does not work with dissimilarity and semi-supervised learning. Suppose, for example, that there are three classes, and that $\mathbf{x}_i, \mathbf{x}_j$ are two unlabeled points whose actual labels are 2 and 3, respectively. Let (i, j) be specified as a dissimilarity edge. In the binary

sub-task of class 1 vs. all other classes, however, this dissimilarity edge should become a similarity edge, since $\mathbf{x}_i, \mathbf{x}_j$ are both in the "rest" meta-class.

2. One-vs-one does not work either. For any particular one-vs-one sub-task (say class 1 vs. 2), it is not clear whether any unlabeled point (say $\mathbf{x}_j$ which actually has class 3) should participate in the one-vs-one semi-supervised learning. If an unlabeled point does not have one of the two labels, its inclusion will likely confuse learning.

3. Using the warped kernel (7.4) in a standard multiclass kernel machine (e.g., multiclass SVM) does not work. Multiclass methods use k discriminant functions $f_1, \cdots, f_k$, one for each class. The warped kernel incorrectly encourages all discriminant functions to honor $f_\cdot(\mathbf{x}_i) + f_\cdot(\mathbf{x}_j) = 0$, which is unnecessary and potentially harmful.

We found all the above approaches indeed hurt accuracy in experiments not reported here.

We therefore need to redesign the multiclass objective in order to incorporate dissimilarity. For simplicity we focus on multiclass SVMs, but our method works for other loss functions, too. There are several formulations of multiclass SVMs (Crammer and Singer, 2002; Lee et al., 2004; Weston and Watkins, 1998). For our purpose it is important to anchor the discriminant functions around zero. For this reason we start with the formulation of Lee et al. (2004). A k-class SVM is defined as the optimization problem of finding functions $\mathbf{f}(\mathbf{x}) = (f_1(\mathbf{x}), \cdots, f_k(\mathbf{x}))$ that solve:

$$\begin{aligned} \min \quad & \tfrac{1}{l} \textstyle\sum_{i=1}^{l} L_{i\cdot}(\mathbf{f}(\mathbf{x}_i) - \mathbf{f}_i)_+ + \lambda \textstyle\sum_{j=1}^{k} \|h_j\|_{\mathcal{H}}^2 \\ \text{s.t.} \quad & \textstyle\sum_{j=1}^{k} f_j(\mathbf{x}_i) = 0, \;\; i = 1 \cdots l, \end{aligned} \tag{7.5}$$

where $f_j(\mathbf{x}) = h_j(\mathbf{x}) + b_j$ for $j = 1 \cdots k$; $h_j \in \mathcal{H}$, which is the RKHS of some kernel K; and $b_j \in \mathbb{R}$. There are l labeled training points. L is an $l \times k$ matrix, with the i-th row $L_{i\cdot} = (1, \cdots, 1, 0, 1, \cdots, 1)$ being an all-one vector except the y_i-th element which is zero. y_i is the given label for $\mathbf{x}_i$. The vector $\mathbf{f}_i = (-1/(k-1), \cdots, 1, -1/(k-1), \cdots)^\top$ is an encoding of the label y_i, where the number 1 occurs in the y_i-th position. The plus function is $(z)_+ = \max(0, z)$. Intuitively, (7.5) means that $\mathbf{f}(\mathbf{x}_i)$ should have elements less than $-1/(k-1)$ for all "wrong classes." It is important to note that the elements of $\mathbf{f}_i$ and $\mathbf{f}(\mathbf{x}_i)$ sum to zero.

We exploit this sum-to-zero label encoding to represent dissimilarity in a convex multiclass SVM objective. To simplify the notation, we will restrict ourselves to dissimilarity edges with weight 1. Similarity edges can be added to the formulation easily by using terms like $(\mathbf{f}(\mathbf{x}_i) - \mathbf{f}(\mathbf{x}_j))^2$ as in previous work (Sindhwani et al., 2005a; Zhu et al., 2003). Given a dissimilarity edge $(s,t) \in \mathcal{D}$, the key idea behind our multiclass dissimilarity formula comes from comparing $\mathbf{f}(\mathbf{x}_s), \mathbf{f}(\mathbf{x}_t)$ for the "good" and "bad" cases. The "good" case is when $\mathbf{f}$ takes the nominal encoding $\mathbf{f}(\mathbf{x}_s) = \mathbf{f}_s$ and $\mathbf{f}(\mathbf{x}_t) = \mathbf{f}_t$, and $\mathbf{f}_s \neq \mathbf{f}_t$. By definition $\mathbf{f}_s$ and $\mathbf{f}_t$ have the form $(-1/(k-1), \cdots, 1, -1/(k-1), \cdots)^\top$, where the elements with value 1 must be at different positions. Hence $\mathbf{f}_s + \mathbf{f}_t$ is a vector with two kinds of elements: $(k-2)/(k-1)$ and $-2/(k-1)$. The "bad" case is when $\mathbf{f}_s = \mathbf{f}_t$, so the elements with value 1 coincide. In this case the sum $\mathbf{f}_s + \mathbf{f}_t$ has two kinds of elements: 2 and $-2/(k-1)$. Comparing "good" and "bad," we do not want any element in $\mathbf{f}(\mathbf{x}_s) + \mathbf{f}(\mathbf{x}_t)$ to be larger than $(k-2)/(k-1)$. We are therefore led to the following dissimilarity objective:

$$\sum_{(s,t)\in\mathcal{D}} \sum_{j=1}^{k} \left(f_j(\mathbf{x}_s) + f_j(\mathbf{x}_t) - \frac{k-2}{k-1} \right)_+^p, \tag{7.6}$$

which is a sum of plus functions raised to the p-th power. The advantages of this definition are that it is convex and simple, and it reduces to our binary SVM dissimilarity formulation when $p = 2, k = 2$.

Following standard practices, one can combine (7.5) and (7.6) into a quadratic program:

$$\begin{aligned} \min \quad & \tfrac{1}{l} \textstyle\sum_{i=1}^{l} L_{i\cdot}(\mathbf{f}(\mathbf{x}_i) - \mathbf{f}_i)_+ + \lambda_1 \sum_{j=1}^{k} \|h_j\|_{\mathcal{H}}^2 \\ & + \tfrac{\lambda_2}{|\mathcal{D}|} \textstyle\sum_{(s,t)\in\mathcal{D}} \sum_{j=1}^{k} \left(f_j(\mathbf{x}_s) + f_j(\mathbf{x}_t) - \frac{k-2}{k-1} \right)_+^2 \\ \text{s.t.} \quad & \textstyle\sum_{j=1}^{k} f_j(\mathbf{x}_i) = 0, \quad i = 1 \cdots n, \end{aligned} \tag{7.7}$$

where n is the sum of the number of unlabeled points that are involved in any dissimilarity edge, plus the number of labeled points l. The representer theorem in Lee et al. (2004) needs to be extended to include these unlabeled points (Zhu and Goldberg, 2006). In particular, the minimizing functions for (7.7) have the form

$$f_j(\mathbf{x}) = \sum_{i=1}^{n} c_{ij} K(\mathbf{x}_i, \mathbf{x}) + b_j \quad \text{for } j = 1, \cdots, k \tag{7.8}$$

The essential difference to supervised learning is that we now have n rather than l representers in (7.8).

We now rewrite the quadratic program in standard form. Note $\|h_j\|^2_{\mathcal{H}} = c_{\cdot j}^\top K c_{\cdot j}$, where $K_{st} = K(\mathbf{x}_s, \mathbf{x}_t)$ is the $n \times n$ Gram matrix. We let $p = 1$ in the dissimilarity objective (7.6). This leads to the primal form

$$\begin{aligned} \min \quad & \tfrac{1}{l}\textstyle\sum_{i=1}^{l} L_{i\cdot}(\mathbf{f}(\mathbf{x}_i) - \mathbf{f}_i)_+ + \lambda_1 \sum_{j=1}^{k} c_{\cdot j}^\top K c_{\cdot j} \\ & + \tfrac{\lambda_2}{|\mathcal{D}|} \textstyle\sum_{(s,t)\in\mathcal{D}} \sum_{j=1}^{k} \left(f_j(\mathbf{x}_s) + f_j(\mathbf{x}_t) - \tfrac{k-2}{k-1}\right)_+ \\ \text{s.t.} \quad & \textstyle\sum_{j=1}^{k} f_j(\mathbf{x}_i) = 0, \quad i = 1 \cdots n. \end{aligned} \tag{7.9}$$

We define an $l \times k$ matrix Y whose i-th row is $\mathbf{f}_i^\top$. Substituting (7.8) into (7.9), we obtain

$$\begin{aligned} \min \quad & \tfrac{1}{l}\textstyle\sum_{i=1\cdots l}^{j=1\cdots k} L_{ij}(K_{i\cdot}c_{\cdot j} + b_j - Y_{ij})_+ \\ & + \lambda_1 \textstyle\sum_{j=1}^{k} c_{\cdot j}^\top K c_{\cdot j} \\ & + \tfrac{\lambda_2}{|\mathcal{D}|} \textstyle\sum_{(s,t)\in\mathcal{D}}^{j=1\cdots k} \left((K_{s\cdot} + K_{t\cdot})c_{\cdot j} + 2b_j - \tfrac{k-2}{k-1}\right)_+ \\ \text{s.t.} \quad & \textstyle\sum_{j=1\cdots k}(K_{i\cdot}c_{\cdot j} + b_j) = 0, \quad i = 1 \cdots n. \end{aligned} \tag{7.10}$$

Finally we introduce an $l \times k$ matrix ξ and a $|\mathcal{D}| \times k$ matrix τ as auxiliary variables. With standard reformulation techniques, we rewrite (7.10) as

$$\begin{aligned} \min \quad & \tfrac{1}{l}\textstyle\sum_{i=1\cdots l}^{j=1\cdots k} L_{ij}\xi_{ij} + \lambda_1 \sum_{j=1}^{k} c_{\cdot j}^\top K c_{\cdot j} \\ & + \tfrac{\lambda_2}{|\mathcal{D}|} \textstyle\sum_{(s,t)\in\mathcal{D}}^{j=1\cdots k} \tau_{stj} \\ \text{s.t.} \quad & K_{i\cdot}c_{\cdot j} + b_j - Y_{ij} \le \xi_{ij}, \quad i = 1 \cdots l, j = 1 \cdots k \\ & \xi_{ij} \ge 0, \quad i = 1 \cdots l, j = 1 \cdots k \\ & (K_{s\cdot} + K_{t\cdot})c_{\cdot j} + 2b_j - \tfrac{k-2}{k-1} \le \tau_{stj}, \\ & \tau_{stj} \ge 0, \quad (s,t) \in \mathcal{D}, j = 1 \cdots k \\ & \textstyle\sum_{j=1\cdots k}(K_{i\cdot}c_{\cdot j} + b_j) = 0, \quad i = 1 \cdots n, \end{aligned} \tag{7.11}$$

where the minimization is over c, b, ξ, τ. The quadratic program has $O(nk)$ variables and constraints.

7.3 Experiments

In the following sections, we empirically demonstrate the benefits of incorporating dissimilarity in several classification tasks.

7.3.1 Standard Binary Datasets

We first experimented using the standard binary datasets g50c and mac-windows (Sindhwani et al., 2005a).[1] The dataset g50c contains 550 examples containing 50 dimensions, and we use $l = 50$ labeled samples. Mac-windows has 1946 examples with 7511 dimensions, also with $l = 50$.

Ideally, we would like to use dissimilarity information based on domain knowledge. However, without such expertise available to us, we performed "oracle experiments" in which we introduce dissimilarity edges between randomly sampled data points with different labels. Because the edges represent ground-truth dissimilarity, we disallow edges to touch labeled points, to prevent the true labels propagating throughout the unlabeled data. Note that the actual label values are not revealed—just the fact that the points should receive *different* label classifications. Simulating domain knowledge in this manner is common for cannot-link clustering and related work. In Section 7.3.3, we present results involving "real" dissimilarity based on domain-specific heuristics.

In this subsection, we introduce dissimilarity in the manifold regularization framework, discussed in Section 7.1. Following Sindhwani et al. (2005a), we start with a Gaussian base kernel K and encode similarity using k-nearest-neighbor graphs with Gaussian weights. Specifically, the weight between kNN points x_i and x_j is $e^{\frac{||x_i - x_j||^2}{2\sigma^2}}$, while all other weights are zero. We then replace some initial similarity edges with dissimilarity edges, as described above, and assign them a relatively large weight (see below) to form the mixed-graph matrix $\mathcal{M}$. Our experiments used the resulting warped kernel K' in both SVM and RLS classifiers. The methods were implemented using LIBSVM and a modified version of the code from Sindhwani et al. (2005a). We used the same parameter values as these authors. These had been tuned in the earlier work with 5-fold cross

[1] Available at `http://vikas.sindhwani.org/manifoldregularization.html`

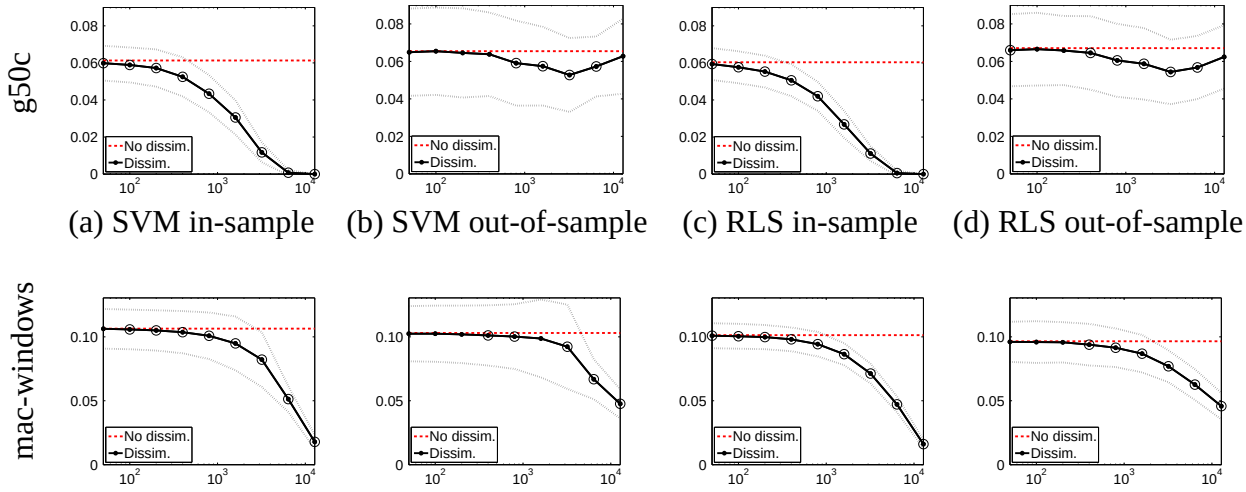
validation using similarity only; our dissimilarity results could become even better with additional parameter tuning.

To compare error rate on unlabeled data used during semi-supervised training, and on new unseen test data, we divided each dataset into four disjoint folds. We then performed 4-fold cross validation, using each fold as a test set once. The test set remains unseen throughout the learning process. The remaining three folds comprised the training set (labeled and unlabeled data). For each train/test split, we trained 10 different classifiers, each time using a different random choice of labeled examples and dissimilarity edges between unlabeled examples. The same random choices are made in all experimental runs, so we can compare results using paired statistical tests. We report classification error rate on the unlabeled training set (*in-sample* performance) and unseen test data (*out-of-sample* performance). Each number is averaged over 4 folds with 10 random trials each. We address two questions in these standard binary dataset experiments:

How does the number of dissimilarity edges influence mean error rate? We experimented first with varying the number of dissimilarity edges in the graph. Since we have high confidence in the oracle edges, we assign each edge a weight equal to the maximal similarity edge weight (close to 1 for our datasets). The baselines here use only similarity edges and are equivalent to LapSVM and LapRLS (Sindhwani et al., 2005a).

Figure 7.1 shows the effect of changing the number of dissimilarity edges in the g50c and mac-windows datasets. Figures 7.1(a,b,e,f) present mean SVM in-sample and out-of-sample error rates using 50–12800 dissimilarity edges, as compared to the baseline with 0 dissimilarity edges. Figures 7.1(c,d,g,h) display comparable results using an RLS classifier. In all plots, we show one standard deviation above and below the error rate curve.

Figure 7.1 shows the positive impact of dissimilarity edges. The effect is greater for in-sample performance; the in-sample points were directly involved in the kernel deformation, so this benefit is to be expected. Our model also generalizes to out-of-sample test data. To measure statistical significance, we performed two-tailed, paired t-tests, comparing the results using each number of dissimilarity edges to the baseline in each of the subplots. The circled settings are statistically significant at the 0.05 level.



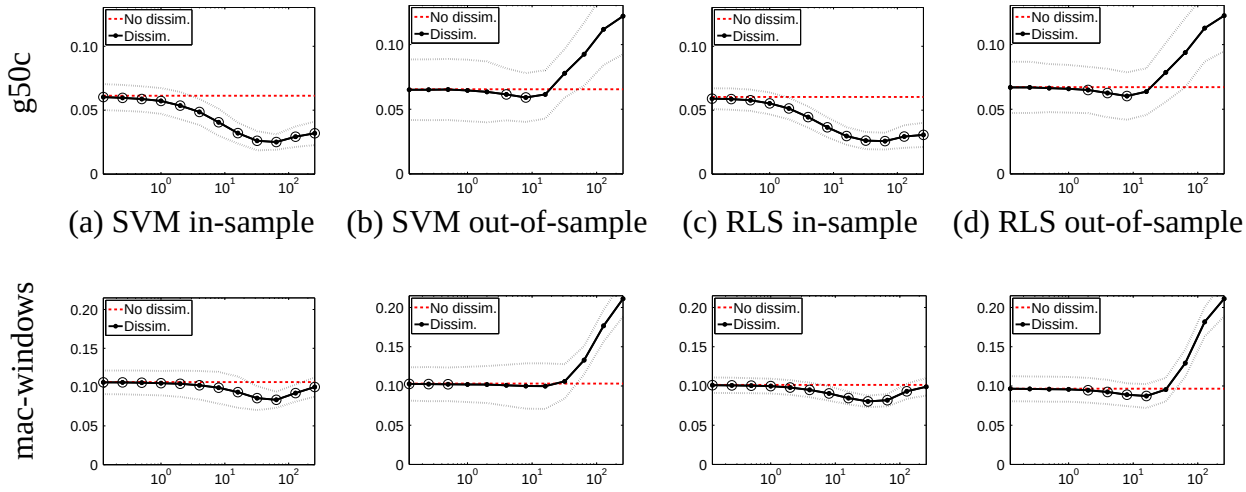
(a) SVM in-sample (b) SVM out-of-sample (c) RLS in-sample (d) RLS out-of-sample

(e) SVM in-sample (f) SVM out-of-sample (g) RLS in-sample (h) RLS out-of-sample

Figure 7.1: Varying the number of dissimilarity edges (x-axis) in the g50c dataset (a-d) and the mac-windows dataset (e-h). y-axis is mean error rate across 4 folds with 10 random trials each. Circled settings are statistically significantly better than the baseline.

While out-of-sample performance steadily improves in the mac-windows dataset (Figures 7.1(f,h)), the g50c out-of-sample error benefits less with 6400 or 12800 dissimilarity edges (Figures 7.1(b,d)). The increase in error rate corresponds with near-zero in-sample error rates, suggesting that the learning algorithm is overfitting the dissimilarity edges. For this small dataset, nearly all of the unlabeled points are touched by one or more of the 6400–12800 dissimilarity edges. (Mac-windows is roughly four times as large, so this is not the case.) It seems the kernel becomes so warped that it fits the g50c unlabeled points perfectly, but becomes less effective in classifying unseen test points. Though we require only $f(\mathbf{x}_i)f(\mathbf{x}_j) < 0$ for $\mathbf{x}_i$ and $\mathbf{x}_j$ to be labeled differently, the dissimilarity terms encourage $f(\mathbf{x}_i) = -f(\mathbf{x}_j)$ for $(i, j) \in \mathcal{D}$. We believe that this unnecessarily stringent requirement is at the root of the observed overfitting when too many dissimilarity terms are included. While the mechanics are still unclear, the inappropriate demand appears to become overwhelming, and generalization error starts to increase.

What is the effect of the weight assigned to dissimilarity edges? In the preceding experiments, we varied the number of dissimilarity edges, but fixed their weights to roughly 1. We next



(a) SVM in-sample (b) SVM out-of-sample (c) RLS in-sample (d) RLS out-of-sample

(e) SVM in-sample (f) SVM out-of-sample (g) RLS in-sample (h) RLS out-of-sample

Figure 7.2: Changing the weight of dissimilarity edges (x-axis) in the g50c dataset (a-d) and the mac-windows dataset (e-h). y-axis is mean error rate across 4 folds with 10 random trials each. Circled settings are statistically significantly better than the baseline.

fixed the number of edges at 200, and experimented with varying this weight by a range of multiplicative factors (Figure 7.2). This effectively places more or less confidence in the dissimilarity edges, compared to the similarity edges. As before, the baseline does not use any dissimilarity.

We observe that in-sample performance tends to benefit from stronger weights on dissimilarity edges (Figures 7.2(a,c,e,g)). The maximal decrease in mean error rate appears at a weight of approximately 64, above which the error rate rises slightly. In both datasets, above a weight of approximately 100, the out-of-sample error rate (Figures 7.2(b,d,f,h)) dramatically rises above the baseline. This appears to be another case of overfitting—the kernel deformation relies too heavily on the dissimilarity edges, and much useful similarity is being ignored. This results in good in-sample performance, at the expense of being able to correctly classify new examples.

7.3.2 Multiclass Handwritten Digit Recognition Dataset

We next experimented with dissimilarity in multiclass classification as described in Section 7.2. We used the standard multiclass dataset *USPS test*, which contains 2007 examples with 256 dimensions, each belonging to one of 10 classes. We used labeled set size $l = 50$. This dataset was also

Dissim.	Overall	In-sample	Out-of-sample
baseline 0	24.48	24.48	24.48
10	24.41	20.47	24.40
20	24.32	23.53	24.33
40	24.27	24.17	24.27
80	**23.96**	23.57	**23.99**
160	**23.63**	24.49	**23.48**
320	**23.30**	23.57	**23.20**

Table 7.1: Mean error rate with varying numbers of dissimilarity edges in the USPS dataset using the multiclass SVM formulation.

used by Sindhwani et al. (2005a) and is available at the URL cited above. We solve the quadratic program in (7.11) using the CPLEX QP solver. We experimented using varying numbers of oracle dissimilarity edges. As before, our dissimilarity edges do not touch labeled points. We consider those examples involved in dissimilarity to be the unlabeled set, and the remaining examples (ignored during training) the unseen test set. We report mean error rates over 10 repeated trials using different random labeled sets and different random unlabeled-unlabeled dissimilarity edges. The λ_1 parameter in (7.11) was optimized using mean test set performance without any dissimilarity. Thus, we are making the baseline as strong as possible. We arbitrarily set $\lambda_2 = 1$. Careful tuning of this parameter could potentially lead to even better results.

Table 7.1 presents the overall, in-sample, and out-of-sample mean error rates using the 2-norm SVM formulation (7.11) with a varying number of dissimilarity edges. Statistically significant reductions in error rate, compared to the baseline, are indicated in bold face. The 2-norm multiclass SVM formulation uses the dissimilarity edges effectively to lower overall and out-of-sample mean error rate for all amounts of dissimilarity edges that we tested.

7.3.3 Predicting Political Affiliation Using Heuristic Dissimilarity Edges

Finally, we experiment with creating real (instead of oracle) dissimilarity edges based on domain knowledge. We experimented with the `politics.com` discussion board text data from

Mullen and Malouf (2006). The task here is to predict the political affiliation of the users posting messages on a political discussion board. We restrict ourselves to the 184 users with *left* (96) and *right* (88) political tendencies. The dataset contains the text of several thousand posts. Quoting behavior is annotated in the dataset, so we know who quoted whom. Since we are interested in classifying each user (as opposed to each post), we concatenated together all posts (excluding quoted text) written by a user. We removed punctuation and common English words, and applied stemming. We then formed *term frequency-inverse document frequency (TF-IDF)* vectors (Manning and Schütze, 1999) for each user using word types occurring 10 or more times, which resulted in 8656 unique terms.

We created dissimilarity edges by the quoting behavior between users. In political discussion boards, users tend to quote posts by users with differing political views (Mullen and Malouf, 2006). For example, users often debate a controversial issue, quoting and disputing each others' previous claims. We declare disagreement between A and B if B quotes A, and the text adjacent to the quoted text contains two or more question marks or exclamation marks, or two or more consecutive words in all capital letters (i.e., Internet shouting[2]). Consider the following illustrative example taken from the current dataset, where the user *Dixie* has quoted and responded to the user *deshrubinator*:

> *deshrubinator*: "You were the one who thought it should be investigated last week."
> *Dixie*: No I didn't, and I made it clear. You are insane! YOU are the one with NO ****ING RESPECT FOR DEMOCRACY!

We create a dissimilarity edge (A,B) if they have exhibited such seemingly hostile behavior toward each other in more than 2 posts. This thresholding ensures that we have seen multiple pieces of evidence for dissimilarity.

It is worth noting that our dissimilarity edges can be created using only simple text processing, and they can be easily defined over unlabeled data (users with unknown political view). For this experiment we do not include similarity edges, since standard text similarity measures will be more sensitive to topics than opinions. Also, unlike the previous "oracle" experiments, here we include

[2]We also require these words to be more than three characters long to avoid false positives from common Internet abbreviations like LOL (laugh out loud).

Classifier	Base error rate	SSL error rate	Δ
SVM	45.67 ± 3.28	40.15 ± 4.95	5.5%
RLS	45.60 ± 3.94	37.99 ± 1.88	7.6%

Table 7.2: Mean error rates for SVM and RLS with and without dissimilarity edges on the politics dataset. Dissimilarity is incorporated through warped kernels. Both differences are statistically significant.

all discovered dissimilarity edges involving labeled and unlabeled data; the only edges discarded are those between two labeled examples. Our scheme is realistic with noisy, "real" edges.

We used a graph of these dissimilarity edges to warp a linear kernel used in SVM and RLS classification. We set the labeled set size $l = 50$ (out of 184) and ran 10 repeated trials with randomly selected labeled examples. On average, 40.7 examples are involved in the dissimilarity edges. Table 7.2 reports the mean error rate on all unlabeled examples for SVM and Regularized Least Squares (RLS) classifiers with ("SSL") and without ("Base") dissimilarity edges. The baseline results use unwarped linear kernels. In both classifiers, we observe a statistically significant reduction in error rate ($p < 0.05$ using a two-tailed, paired t-test); it appears that the "real-world" dissimilarity edges aid classification. Upon closer inspection, however, we also notice the improvement comes mostly from in-sample error reduction, and it does not generalize as well to out-of-sample data like in previous experiments. We suspect this could be due to the high initial error rate.

Finally, as a post-experiment study, we investigated how many of our heuristically derived dissimilarity edges were actually consistent with the true labels. It turns out that 85 out of the 103 edges (83%) are in fact "true" dissimilarity edges. Thus, we have shown that, even if 17% of the dissimilarity edges represent false domain knowledge, we can achieve a significant improvement in overall error rate.

7.4 Conclusions

We presented a convex algorithm to encode dissimilarity in semi-supervised learning. We demonstrated that when such dissimilarity domain knowledge is available, our algorithm can take advantage of it and improve classification. The major advantage of our dissimilarity-encoding formulations for binary and multiclass classification is convexity. However, they probably specify the relation between the discriminant function f at dissimilarity samples $\mathbf{x}_i$ and $\mathbf{x}_j$ more than necessary. For example, in the binary case we prefer $f(\mathbf{x}_i) = -f(\mathbf{x}_j)$, while ideally it is sufficient to require $f(\mathbf{x}_i), f(\mathbf{x}_j)$ having opposite signs. Finding computationally efficient encodings for this sufficient condition is a direction for future research.

Chapter 8

Regularization with Order Preferences

As in the preceding chapter, we now explore a new semi-supervised learning algorithm using a novel regularizer Ω_{SSL}. In particular, we introduce a new regularizer for semi-supervised kernel regression that encodes "order preferences"—beliefs about the relative order of the target values for a pair of unlabeled instances (Zhu and Goldberg, 2007).

As a motivating example, consider the task of predicting real estate prices. The price of a house varies significantly depending on its location and many other factors. However, a domain expert may determine that, everything else being "roughly equal," the feature *number of bedrooms* determines the order of house prices. For instance, a 4-bedroom house is more expensive than a 3-bedroom one.

At first glance, it may appear that such knowledge can be enforced by a positive correlation between the feature and the target. However, modeling such knowledge as positive correlation can be difficult in *non-linear* kernel regression, because of the non-linear feature mapping. Besides, in general a correlation may only hold for part of the range of the feature value, and it would be inappropriate to force the correlation across the range. We would like a more general approach to capture such knowledge.

Our method encodes such domain knowledge with *order preferences* on unlabeled examples. That is, for all pairs of unlabeled examples x_i, x_j satisfying the "roughly equal" condition, we encode domain knowledge specifying the *order* between their target values $f(x_i)$ and $f(x_j)$, even though their actual target values are unknown. Respecting the domain knowledge amounts to incorporating the order preferences into a kernel regression framework. When labeled data is scarce, these order preferences should improve our regression model.

Another practical application of our approach is in predicting Internet file transfer rates based on network properties like round trip time, available bandwidth, queuing delay, package loss rate, and so on (Mizra et al., 2007). The features have intuitive impact on transfer rate, but the exact relation is highly non-linear and unknown. We can, however, easily create (noisy) order preferences on unlabeled data using domain knowledge. In general, order preferences can encode certain complex domain knowledge.

In the next few sections, we formulate the problem of learning with order preferences as a linear program that can be solved efficiently. Experiments on benchmark datasets, sentiment analysis, and housing price problems show that the proposed algorithm outperforms standard regression, even when the order preferences are noisy.

8.1 Regression with Order Preferences

Let us formally define our regression problem. In addition to a labeled training set $\{(x_i, y_i)\}_{i=1}^{l}$, we assume that we are given p order preferences between pairs of unlabeled examples. An order preference is defined by a tuple (i, j, d, w), with the interpretation that we would like $f(x_i) - f(x_j) \geq d$. As discussed below, we encode it as a soft preference rather than a hard constraint. The scalar $w \geq 0$ is the weight (confidence) for the preference.

Obviously knowing the order preferences is much weaker than knowing the labels of the unlabeled examples. In this sense the preferences are a form of weakly labeled data or side information. We would like to use them to improve regression.

It is possible to represent order preferences as directed edges in a graph (Dekel et al., 2003), where the edges represent asymmetric order information. However, it is worth noting that order preferences can also encode similarity. For example, the two preferences $(i, j, 0, w), (j, i, 0, w)$ encode $f(x_i) = f(x_j)$. More generally, the two preferences $(i, j, -\epsilon, w), (j, i, -\epsilon, w)$ encode closeness: $|f(x_i) - f(x_j)| \leq \epsilon$. It is also easy to encode $a \leq f(x_i) - f(x_j) \leq b$. As special cases of order preferences, one can also encode unary preferences $f(x_i) \leq g(x_i)$, $f(x_i) = g(x_i)$, or $f(x_i) \geq g(x_i)$, where g is some given function. The unary preferences are closely related to the work of Mangasarian et al. (2004), which adds them to kernel machines.

Our approach to add order preferences to kernel regression is to treat them as an additional form of regularization. The standard risk minimization framework for kernel regression is

$$\min_{\mathbf{f}\in\mathcal{H}} \quad \sum_{i=1}^{l} c(x_i, y_i, f(x_i)) + \lambda\Omega(\|\mathbf{f}\|_{\mathcal{H}}), \tag{8.1}$$

where $\mathcal{H}$ is the Reproducing Kernel Hilbert Space (RKHS) induced by some kernel, $c()$ is a loss function for regression, λ is a weight parameter on the regularizer, and $\Omega()$ is a monotonic increasing function.

We now define an additional regularization term $r(x, f)$ based on the order preferences, which plays the role of Ω_{SSL} discussed earlier. Intuitively if the function $\mathbf{f}$ satisfies all order preferences, r should be zero; if $\mathbf{f}$ violates some, r increases. A natural choice is to use a shifted hinge function: for order preference (i, j, d, w), the regularization term for this single preference is $w\max(d - (f(x_i) - f(x_j)), 0)$. That is, it is zero if the preference is satisfied; otherwise it is the amount the preference is violated, weighted by w. As a side note, we point out that if we have two preferences $(i, j, -\epsilon, w), (j, i, -\epsilon, w)$, this would form the ϵ-insensitive loss (Smola and Schölkopf, 2004).

We define the regularization term $r(x, f)$ as the sum of shifted hinge functions on all order preferences:

$$r(x, f) = \sum_{q=1}^{p} w_q \max(d_q - (f(x_{iq}) - f(x_{jq})), 0). \tag{8.2}$$

We note that order preferences have been used in ranking problems (Herbrich et al., 2000; Burges et al., 2005; Yu et al., 2006; Chu and Ghahramani, 2005); in particular Joachims (2002) employed a similar shifted hinge function for ranking. However, they have not been used in regression before. In the end, our high-level optimization problem is

$$\min_{\mathbf{f}\in\mathcal{H}} \quad \sum_{i=1}^{l} c(x_i, y_i, f(x_i)) + \lambda_1\Omega(\|\mathbf{f}\|_{\mathcal{H}}) + \lambda_2 r(x, f). \tag{8.3}$$

8.2 A Linear Program Formulation

To fully specify the above problem, we choose to use the ϵ-insensitive loss $c(x, y, f) = |y - f|_\epsilon$ in support vector regression:

$$|y - f|_\epsilon = \begin{cases} 0 & \text{if } |y - f| \leq \epsilon \\ |y - f| - \epsilon & \text{otherwise.} \end{cases} \tag{8.4}$$

We further choose $\Omega(\|f\|_\mathcal{H})$ to be a linear function, in this case the 1-norm of the dual parameters discussed below, resulting in 1-norm support vector machines (Bradley and Mangasarian, 1998; Bi et al., 2003; Zhu et al., 2004a). The formulation originates from generalized support vector machines (Mangasarian, 2000). Such 1-norm support vector machines are comparable in performance to the standard 2-norm support vector machines, but with the advantage that they can be solved as *linear programs*, which tends to be more efficient.

The solution can be characterized by a representer theorem (Kimeldorf and Wahba, 1971; Schölkopf et al., 2001): The minimizer $\mathbf{f}^* \in \mathcal{H}$ admits the form $f^*(x) = \sum_{i=1}^{l+2p} \alpha_i K(x_i, x)$, where x_i ranges from the labeled examples to the unlabeled examples involved in the p order preferences. The proof uses the standard orthogonality argument, and is omitted for space consideration.

Let $K(x, \mathbf{x}_{1:l})$ denote the row vector of kernel values between a point x and the labeled data $\mathbf{x}_{1:l}$. We represent our function $\mathbf{f}$ in dual form by

$$f(x) = K(x, \mathbf{x}_{l:l})\alpha + \alpha_0 \tag{8.5}$$

where α is a column vector of dual parameters, one for each labeled point; α_0 is a bias scalar. This amounts to approximating the representer theorem by setting dual parameters not on the labeled data to zero for a sparse representation. Our linear-program regression problem is

$$\begin{aligned} \min_{\alpha,\alpha_0} \quad & \tfrac{1}{l} \textstyle\sum_{i=1}^{l} |y_i - f(x_i)|_\epsilon + \lambda_1 \|\alpha\|_1 + \\ & \lambda_2 \tfrac{1}{p} \textstyle\sum_{q=1}^{p} w_q \max(d_q - (f(x_{iq}) - f(x_{jq})), 0), \end{aligned} \tag{8.6}$$

where $\|\alpha\|_1 = \sum_{i=1}^{l} |\alpha_i|$ is the 1-norm of α. The bias α_0 is not regularized.

We transform (8.6) into a standard linear program by introducing auxiliary variables for the three terms respectively. Let $\mathbf{1}$ be the all-one vector, ξ an l-vector of slack variables, η an l-vector,

ν a p-vector, $\mathbf{d}$ the difference vector, $\mathbf{w}$ the weight vector, $K(\mathbf{x}^i_{1:p}, \mathbf{x}_{1:l})$ the $p \times l$ kernel matrix between the first points in the order constraints and the labeled data, and $K(\mathbf{x}^j_{1:p}, \mathbf{x}_{1:l})$ the same sized kernel matrix between the second points in the order constraints and the labeled data. Vector inequalities are element-wise. With standard transform techniques, our linear program for kernel regression with order preferences can be written as:

$$\begin{aligned} &\min_{\alpha,\alpha_0,\xi,\eta,\nu} \quad \tfrac{1}{l}\mathbf{1}^\top\xi + \lambda_1\mathbf{1}^\top\eta + \tfrac{\lambda_2}{p}\mathbf{w}^\top\nu \\ &\text{s.t.} -\xi - \epsilon\mathbf{1} \le \mathbf{f}_{1:l} - K(\mathbf{x}_{1:l}, \mathbf{x}_{1:l})\alpha - \alpha_0\mathbf{1} \le \xi + \epsilon\mathbf{1} \\ &\quad \xi \ge 0 \\ &\quad -\eta \le \alpha \le \eta \\ &\quad (K(\mathbf{x}^i_{1:p}, \mathbf{x}_{1:l}) - K(\mathbf{x}^j_{1:p}, \mathbf{x}_{1:l}))\alpha \ge \mathbf{d} - \nu \\ &\quad \nu \ge 0. \end{aligned} \tag{8.7}$$

This is a linear program with $3l + p + 1$ variables and $5l + 2p$ constraints. The global optimal solution can be found efficiently.

As noted above, our order preferences comprise another unlabeled-data-dependent regularizer Ω_{SSL}, like that of manifold regularization or S3VMs. These methods and our order preferences all encode some domain knowledge other than labels. One might establish many order preferences automatically generated by applying heuristics to the unlabeled data. For example, the fact that higher bandwidth, shorter delay and less package loss tend to promote higher file transfer rates, could be used to supply a large number of terms in the regularizer. Our order preferences may contain slightly stronger information than labels, and we view them as filling in the continuum between supervised learning and semi-supervised learning. Note that it is possible to combine order preferences with existing semi-supervised learning methods by adding the respective regularizer terms together (with appropriate weights) to form a new regularizer.

8.3 Experiments

We demonstrate the benefit of order preferences with four groups of experiments. We implemented our linear program (8.7) using CPLEX. All experiments ran quickly. Solving the LP for each trial takes 0.2 to 0.5 seconds depending on the number of order preferences and unlabeled

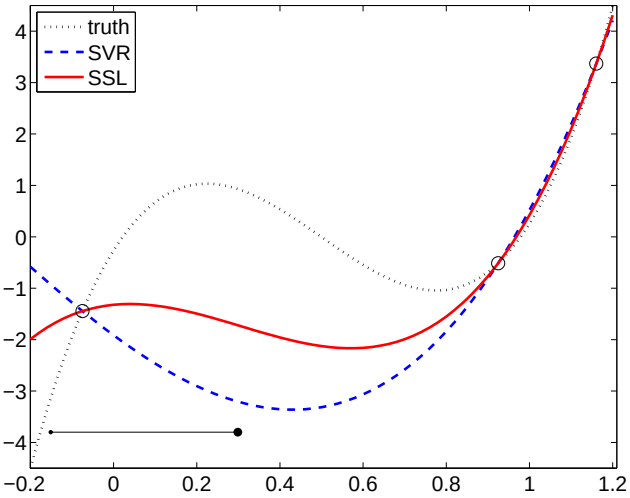
data size. In all experiments, ϵ in the ϵ-insensitive loss (8.4) was set to 0, and preference weights $\mathbf{w}$ were set to 1. We use the acronym SSL for (8.7), and SVR for the corresponding standard 1-norm support vector regression (i.e., $\lambda_2 = 0$). We also experimented with standard 2-norm support vector regression using SVM$^{\text{light}}$ (Joachims, 1999a), and the results were comparable to SVR and not reported here. Since our focus is on the effect of order preference in improving SVR, we will use SVR as our baseline in the experiments.

8.3.1 A Toy Example

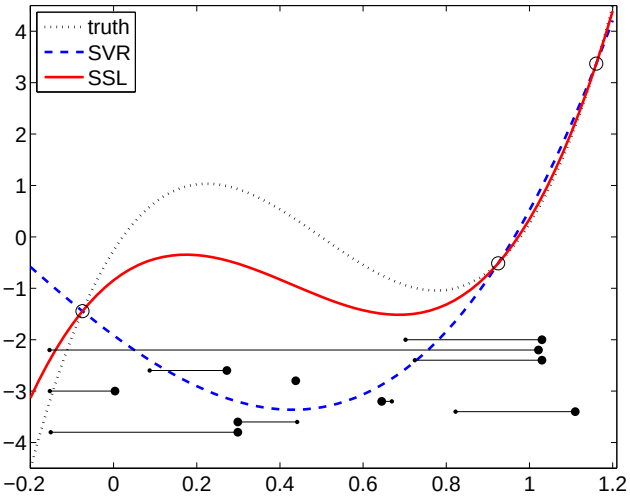
First we use a toy example to illustrate order preferences. We constructed a polynomial function of degree 3 as our target (the dotted line in Figure 8.1(a)). We randomly sampled three points (the open circles) from the target function as training data and gave them to SVR. For this experiment we used a linear kernel and set $\lambda_1 = 0$. Since there were not enough training data points, SVR produced a fit (the dashed line) through the training points but very different from the target.

We then randomly selected a pair of unlabeled points $-0.15, 0.30$. Note they did not coincide with the training points. Without revealing the actual target values at these points, we constructed an order preference using their true order: $(0.30, -0.15, 0, 1)$, or equivalently $f(0.30) - f(-0.15) \geq 0$. Note we set $d = 0$ so that the order preference specified their order but not the true difference; hence it was weaker. We set $w = 1$. In Figure 8.1(a) the order preference is shown at the lower left as a line linking the two unlabeled points (black dots). The point with the larger value has a larger dot. SVR happened to violate the order preference. With the three training points and this order preference, SSL produced a better fit (the solid line).

In Figure 8.1(b) we added more order preferences, generated similarly from random unlabeled point pairs and their true order. Note some preferences were already satisfied by SVR. The SSL function was further improved. We consistently observed such behavior in repeated random trials.



(a) one order preference



(b) ten order preferences

Figure 8.1: A toy example comparing SVR and SSL, showing the benefit of order preferences.

8.3.2 Benchmark Datasets

We experimented with five regression benchmark datasets (Boston, Abalone, Computer, California, Census),[1] and report results on all of them. One difficulty in working with such standard datasets is creating sensible order preferences on unlabeled data. Ideally the order preferences would be prepared by experts with domain knowledge on the tasks. Lacking such experts, we had

[1] Available at `http://www.niaad.liacc.up.pt/~ltorgo/Regression/DataSets.html`.

to create simulated order preferences from the relation of true values on unlabeled points (more details later; Note, however, we never give out the true values themselves). Therefore our results on benchmark datasets should be viewed as "oracle experiments." Nonetheless they are useful indications of how well our regression would perform given such domain knowledge.

For each benchmark dataset, we normalized its input features to zero mean, unit variance. For categorical features with k distinct values, we mapped them into indicator vectors of length k. We used Radial Basis Function (RBF) kernels $k(x, x') = \exp(-\sigma\|x - x'\|^2)$ for all datasets. We used 5-fold cross validation to find the optimal RBF bandwidth σ, and SVR 1-norm weight λ_1. The parameters were tuned for SVR on a 9×9 logarithmic grid in $10^{-4} \leq \sigma \leq 10^4$ and $10^{-4} \leq \lambda_1 \leq 10^4$. We simply fixed λ_2 at 1. This is partly justified by the fact that in (8.6), the 'shifted hinge function' is on a similar scale to the ϵ-insensitive loss; both incur a linear penalty when violated. Tuning λ_2 might produce better results than reported here, but with limited labeled data (which has been used to tune λ_1 and σ for SVR already) it is hard to do.

All experiments were repeated for 20 random trials. Different algorithms shared the same random trials so we could perform paired statistical tests. In each trial we split the data into three parts: l labeled points, u unlabeled points that were used to generate order preferences, and test points that were the rest of the dataset (see Table 8.1 Partition). Test points were unseen by either algorithm during training. All results we report are test-set mean-absolute-error over the 20 trials. Let t be the test set size. Test-set mean-absolute-error is defined as $\sum_{i \in \text{test}} |y_i - f(x_i)|/t$. We address the following questions:

Can order preferences improve regression? We randomly sampled with replacement $p = 1000$ pairs (x_i, x_j) from the u unlabeled points. For each sampled pair, we generated an order preference from the true target values y_i, y_j. Without loss of generality let $y_i \geq y_j$. Our simulated order preference was

$$f(x_i) - f(x_j) \geq 0.5(y_i - y_j). \tag{8.8}$$

Let us explain our order preferences. We could have created the 'perfect' order preferences with the pair: $f(x_i) - f(x_j) \geq y_i - y_j$ and $f(x_i) - f(x_j) \leq y_i - y_j$. They together encode $f(x_i) - f(x_j) = y_i - y_j$. But in real tasks it might be difficult to know the exact difference $y_i - y_j$, so we did not do

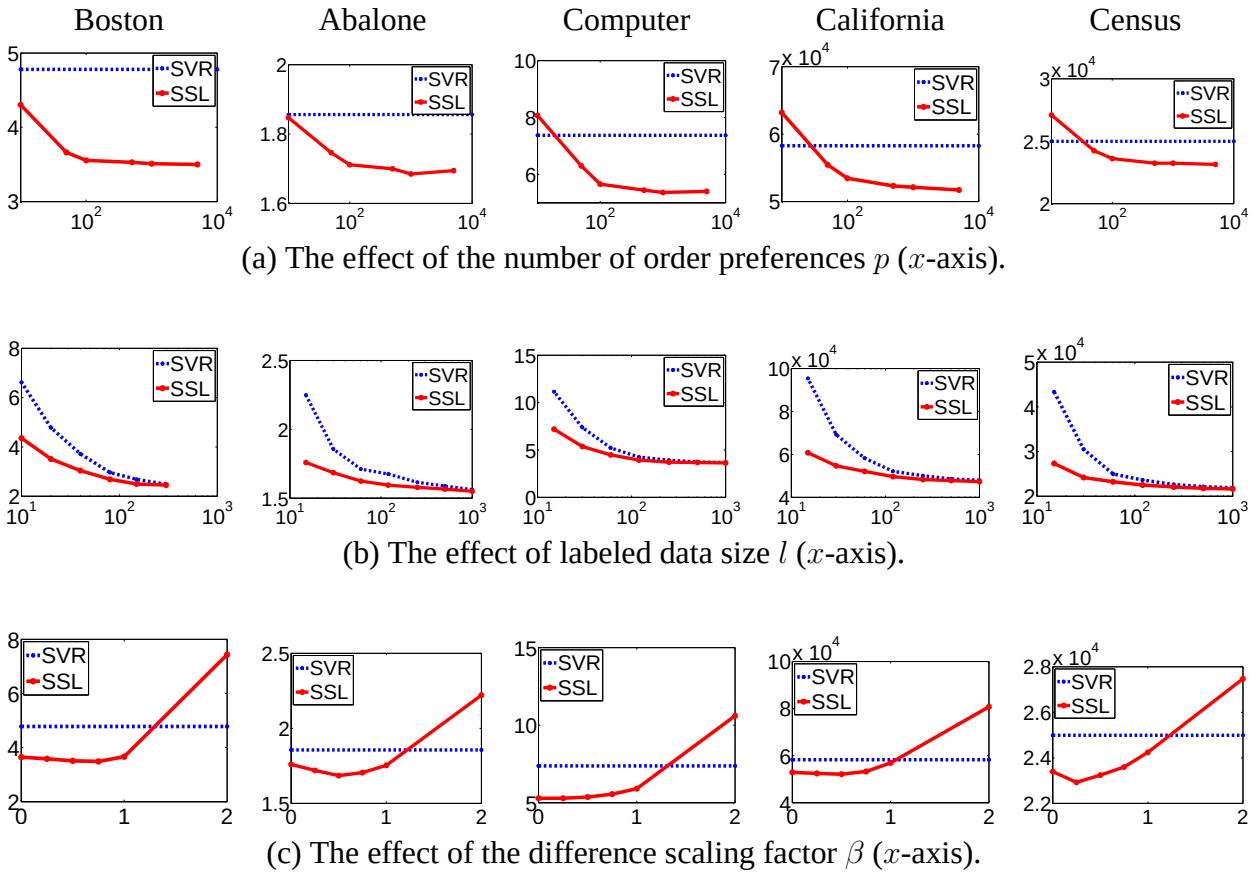
Dataset	Partition		Mean absolute error		Improvement
	dim	$l/u/$test	SVR	SSL	
Boston	13	20/200/286	4.780 ± 1.351	$\mathbf{3.511 \pm 0.376}$	27%
Abalone	8	30/1000/3147	1.856 ± 0.180	$\mathbf{1.685 \pm 0.102}$	9%
Computer	21	30/1000/7162	7.373 ± 3.445	$\mathbf{5.364 \pm 0.998}$	27%
California	8	60/1000/19580	58268 ± 4435	$\mathbf{52120 \pm 1843}$	11%
Census	16	60/1000/21724	24992 ± 1377	$\mathbf{23241 \pm 901}$	7%

Table 8.1: Benchmark data. 'dim' is the dimension of input features; $l/u/$test are labeled, unlabeled and test set sizes respectively. SVR is 1-norm support vector regression. SSL is semi-supervised regression with 1000 random order preferences sampled from u. The results are test-set mean-absolute-error and standard deviation over 20 random splits. All differences between SVR and SSL are significant with a paired t-test at the 0.01 level.

that. On the other hand, with inequality preferences we could have set $f(x_i) - f(x_j) \geq 0$. It would only encode order, without any information on the actual difference. But in real tasks one might have some rough estimate of the difference, and (8.8) was meant to simulate this estimate. Table 8.1 compares the test-set mean-absolute-error of SVR and SSL. The differences on all datasets are significant with a paired t-test at the 0.01 level. We conclude that, with the order preferences (8.8), SSL significantly improves regression performance over SVR.

What if we change the number of order preferences p? One expects a larger gain with more order preferences p. We systematically varied p from 10 to 5000, keeping everything else the same as in Table 8.1. Figure 8.2(a) shows that it was indeed the case. A very small p sometimes hurts SSL, making it worse than SVR. But as p grows larger SSL rapidly improves, and levels off at around $p = 100$. This indicates that one needs only a moderate amount of order preferences to enjoy the benefit.

What if we change the labeled data size l? The benefit of order preferences is expected to diminish with more labeled data. We fixed the number of order preferences $p = 1000$, and systematically varied l. As expected, Figure 8.2(b) shows that SSL is most useful when l is small, and the benefit reduces as l grows.



(a) The effect of the number of order preferences p (x-axis).

(b) The effect of labeled data size l (x-axis).

(c) The effect of the difference scaling factor β (x-axis).

Figure 8.2: The effect of various parameters on SSL on the Benchmark data. y-axis is test-set mean-absolute-error.

How precise do the order preferences need to be? Extending (8.8), one can define order preferences as $f(x_i) - f(x_j) \geq \beta(y_i - y_j)$ where β controls how precise they are. As mentioned earlier, $\beta = 0$ only supplies order information, and a larger β estimates the differences. We varied β from 0 to 2 (over-estimate) for the experiments in Table 8.1. Figure 8.2(c) shows that with only the order ($\beta = 0$) SSL already outperformed SVR. With a conservative estimate of the differences ($0 < \beta < 1$) SSL was even better. However larger β seems to be inferior. This might be advantageous in practice, since one does not need to know the precise differences, and can err on the safe side.

8.3.3 Sentiment Analysis in Movie Reviews

We next experimented with the real-world problem of sentiment analysis in movie reviews. Given a movie review text document x, we would like to predict $f(x)$, the rating (e.g., '4 stars') given to the movie by the reviewer. We assume that by looking at the wording of unlabeled reviews, one can determine that some movies will likely be rated higher than others (even though we do not know their actual ratings). These are incorporated as order preferences. We worked on the "scale dataset v1.0" with continuous ratings,[2] which was prepared and first used by Pang and Lee (2005). It contains four authors with 1770, 902, 1307, 1027 reviews respectively. For each author, we varied $l \in \{30, 60, 120\}$, and let $u = 500, p = 500$. The remaining reviews were test examples. Each experiment was repeated for 20 random trials. All reported results are test-set mean-absolute-error. Each review document was represented as a word-presence vector, normalized to sum to 1. We used a linear kernel, set $\lambda_1 = 10^{-7}$ and $\lambda_2 = 1$.

As a proxy for expert knowledge, we used a completely separate "snippet dataset" also located at the above URL. The snippet dataset is very different from the scale dataset: it contains single punch line sentences (snippets) instead of full reviews; the snippets have binary positive/negative labels instead of continuous ratings; it comes from different authors on different movies. We trained a standard binary, linear-kernel SVM classifier g on the *snippet* data using SVM$^{\text{light}}$. We then applied g on random pairs of unlabeled movie reviews x_i, x_j in the *scale* dataset. The order of the continuous margin output $g(x_i), g(x_j)$ serves as our proxy for expert knowledge.[3] Since this is a very crude and noisy estimate, we created an order preference $(i, j, 0, 1)$ only if $g(x_i) - g(x_j) > 0.25$, where 0.25 is an arbitrary threshold. Note we set $d = 0$ since we do not know the difference in rating. Table 8.2 presents the results of our sentiment analysis experiments. As expected, SSL is most useful when l is small, and the gain over SVR gradually diminishes with larger l. SSL leads to improvements in all cases, and the differences are significant (bold) with paired t-tests at the

[2] Available at `http://www.cs.cornell.edu/people/pabo/movie-review-data/`.

[3] Our use of g simulates a layman (not an expert) reading two reviews and saying "the author liked this one more than that one." This layman does not have enough experience to predict the actual star ratings, but is able to tell that one sounds more positive than the other.

Dataset	l/u/test	SVR	SSL	Improvement
Author (a)	30/500/1240	0.1383 ± 0.0072	0.1362 ± 0.0028	1.5%
	60/500/1210	0.1323 ± 0.0042	0.1311 ± 0.0025	0.9%
	120/500/1150	0.1224 ± 0.0042	0.1219 ± 0.0024	0.4%
Author (b)	30/500/372	0.1645 ± 0.0146	$\mathbf{0.1540 \pm 0.0046}$	6.4%
	60/500/342	0.1514 ± 0.0063	$\mathbf{0.1496 \pm 0.0046}$	1.2%
	120/500/282	0.1431 ± 0.0063	$\mathbf{0.1416 \pm 0.0062}$	1.0%
Author (c)	30/500/777	0.1405 ± 0.0163	0.1357 ± 0.0070	3.4%
	60/500/747	0.1268 ± 0.0072	0.1258 ± 0.0038	0.8%
	120/500/687	0.1150 ± 0.0048	0.1138 ± 0.0047	1.0%
Author (d)	30/500/497	0.1433 ± 0.0151	$\mathbf{0.1350 \pm 0.0052}$	5.8%
	60/500/467	0.1366 ± 0.0104	$\mathbf{0.1293 \pm 0.0037}$	5.3%
	120/500/407	0.1256 ± 0.0092	0.1226 ± 0.0038	2.4%

Table 8.2: Movie review sentiment analysis mean-absolute-error for each author. Statistically significant improvements by SSL are highlighted in bold.

0.05 level in about half of the cases.[4] We expect better order preferences from advanced natural language processing (e.g., parsing) to bring larger improvements.

8.3.4 Predicting Housing Prices Using Heuristic Order Preferences

As a final real-world experiment, we played the role of real estate experts to carry out the scenario introduced in the beginning of the chapter. We used the same California dataset in Table 8.1, but this time with order preferences derived from domain knowledge instead of oracles.

The task is to predict the median house value for 20640 groups of houses throughout the state. With other factors being roughly equal, we believe the value is largely determined by the number of bedrooms. We decided that two groups are "roughly equal" if they are located within 25 miles of each other (i.e., they are in the same community), their median house ages differ by at most 10 years, and they are inhabited by residents whose median income level differs by at most $1000.

We repeated the experimental setup in the benchmark section, and for each random trial, we created approximately 1200 order preferences. Specifically, for all pairs of housing groups in the

[4] As a sanity check, we also experimented with *wrong* order preferences by intentionally flipping all preferences $(i, j, 0, 1)$ into $(j, i, 0, 1)$. As expected, SSL with wrong order preferences became *worse* than SVR by 1% – 13% for different authors at $l = 120$.

Dataset	Partition		Mean absolute error		Improvement
	dim	$l/u/$test	SVR	SSL	
California	8	60/1000/19580	58268 ± 4435	$\mathbf{54664 \pm 2521}$	6%

Table 8.3: Using "real-world" order preferences generated from domain knowledge. The improvement is statistically significant.

labeled and unlabeled data that satisfy the "roughly equal" criteria, we created a preference that the group with more bedrooms has a higher target value. We omitted preferences between two labeled groups, since they are either redundant or incorrect. We set $w = 1$ and $d = 0$, and used the same λ parameters as in the benchmark section. Note that the order preferences are created without any knowledge of the actual target values, and that the relations we constructed are highly non-linear.

As seen in Table 8.3, the heuristic preferences led to a 6% reduction in test-set mean-absolute-error in SSL (54664 ± 2521) compared to SVR (58268 ± 4435). The difference is statistically significant with a paired t-test at the 0.01 level.

This experiment demonstrates that order preferences with some noise can still be beneficial. In fact, a post-experimental analysis of the created order preferences revealed that only 70% were actually accurate (i.e., 30% of "roughly equal" housing group pairs do *not* have the predicted relation based on bedrooms). We expect our method to extend well to new tasks (e.g., predicting Internet file transfer rates) where large numbers of reasonably accurate order preferences can be generated automatically.

8.4 Conclusions

We presented a novel semi-supervised kernel regression algorithm with order preferences, formulated as a linear program. We showed that even with noisy, heuristic order preferences, the regression performance is improved. Our algorithm can be easily extended beyond regression. For example, one future direction is to apply order preferences to ordinal classification (Chu and Keerthi, 2005).

Chapter 9

Graph-Based Semi-Supervised Learning for Sentiment Categorization

Finally, we conclude Part III by considering the application of graph-based semi-supervised learning to a problem in the natural language processing area of sentiment analysis (Goldberg and Zhu, 2006). Sentiment analysis of text documents has received considerable attention recently; see the recent survey by Pang and Lee (2008). Unlike traditional text categorization based on topics, sentiment analysis attempts to identify the subjective sentiment expressed (or implied) in documents, such as consumer product reviews, movie reviews, or even online discussion about politics. This line of research is becoming increasing popular due to its high potential impact on consumers, product manufacturers, the entertainment industry, and government.

In this work, we specifically address the sentiment analysis task of rating inference proposed by Pang and Lee (2005). Given a set of documents (e.g., movie reviews) and accompanying ratings (e.g., "4 stars"), the task calls for inferring numerical "star" ratings for unlabeled documents based on the perceived sentiment expressed by their text. In particular, we are interested in the transductive setting where labeled data is scarce, but we already have access to the unlabeled reviews for which we care to make predictions. Labeled data may come from a structured review site where review authors are required to assign a star rating. However, the Web is filled with blog posts offering opinions on movies or products. Such posts rarely contain a numerical rating, but it may be useful in practice to be able to assign such a number for analysis of public opinion as a whole.

In this work, we demonstrate that unlabeled reviews can significantly improve rating-inference performance. This chapter contains three contributions:

- We apply graph-based semi-supervised learning to the novel domain of sentiment analysis;
- We design a special graph that encodes our assumptions for rating-inference problems, including:
  1. Similar reviews tend to have similar ratings;
  2. The ratings of labeled reviews should be respected;
  3. If another learning algorithm exists for the task, that algorithm's rating predictions should be considered, too;
  4. We have different confidence levels in labeled and unlabeled data.

  The graph is discussed in Section 9.1, and the associated optimization problem in Section 9.2;
- We show the benefit of semi-supervised learning for rating inference with extensive experimental results in Section 9.3.

The semi-supervised rating-inference problem is formalized as follows. There are n review documents $x_1 \dots x_n$, each represented by some standard feature representation (e.g., word-presence vectors). Without loss of generality, let the first $l \leq n$ documents be labeled with ratings $y_1 \dots y_l \in C$. The remaining documents are unlabeled (transductive setting). The set of numerical ratings are $C = \{c_1, \dots, c_C\}$, with $c_1 < \dots < c_C \in \mathbb{R}$. For example, a one-star to four-star movie rating system has $C = \{0, 1, 2, 3\}$. We seek a function $f : x \mapsto \mathbb{R}$ that gives a continuous rating $f(x)$ to a document x. Classification is done by mapping $f(x)$ to the nearest discrete rating in C. Note this is ordinal classification, which differs from standard multi-class classification in that C is endowed with an order. In the following we use "review" and "document," "rating" and "label" interchangeably.

We make two assumptions:

1. We are given a *similarity measure* $w_{ij} \geq 0$ between documents x_i and x_j. w_{ij} should be computable from features, so that we can measure similarities between any documents, including unlabeled ones. A large w_{ij} implies that the two documents tend to express the

same sentiment (i.e., rating). We experiment with *positive-sentence percentage* (PSP) based similarity (Pang and Lee, 2005) and mutual-information modulated word-vector cosine similarity. Details can be found in Section 9.3.

2. Optionally, we are given numerical rating predictions $\hat{y}_{l+1}, \ldots, \hat{y}_n$ on the unlabeled documents from a separate learner, for instance ϵ-insensitive support vector regression (Joachims, 1999a; Smola and Schölkopf, 2004), as used by Pang and Lee (2005). This acts as an extra knowledge source for our semi-supervised learning framework to improve upon. We note our framework is general and works without the separate learner, too.

9.1 A Graph for Sentiment Categorization

We now describe our graph for the semi-supervised rating-inference problem. We do this piece by piece with reference to Figure 9.1. Our undirected graph $G = (V, E)$ has $2n$ nodes V, and weighted edges E among some of the nodes.

- Each document is a node in the graph (open circles, e.g., x_i and x_j). The true ratings of these nodes $f(x)$ are unobserved. This is true even for the labeled documents because we allow for noisy labels. Our goal is to infer $f(x)$ for the unlabeled documents.

- Each labeled document (e.g., x_j) is connected to an observed node (dark circle) whose value is the given rating y_j. The observed node is a "dongle" since it only connects to x_j. As we point out later, this serves to pull $f(x_j)$ towards y_j. The edge weight between a labeled document and its dongle is a large number M. M represents the influence of y_j: if $M \to \infty$ then $f(x_j) = y_j$ becomes a hard constraint.

- Similarly each unlabeled document (e.g., x_i) is also connected to an observed dongle node $\hat{y}_i$, whose value is the prediction of the separate learner. Therefore we also require that $f(x_i)$ is close to $\hat{y}_i$. This is a way to incorporate multiple learners in general. We set the weight between an unlabeled node and its dongle arbitrarily to 1 (the weights are scale-invariant

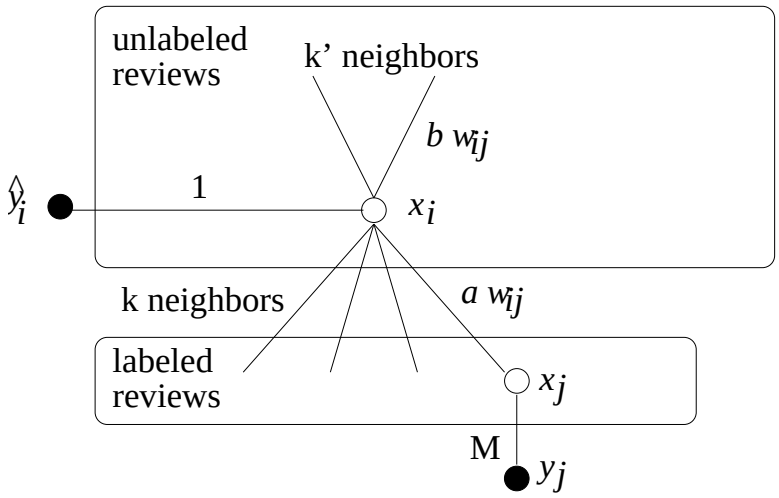


Figure 9.1: The graph for semi-supervised rating inference.

otherwise). As noted earlier, the separate learner is optional: we can remove it and still carry out graph-based semi-supervised learning.

- Each unlabeled document x_i is connected to $kNN_L(i)$, its k nearest *labeled documents*. Distance is measured by the given similarity measure w. We want $f(x_i)$ to be consistent with its similar labeled documents. The weight between x_i and $x_j \in kNN_L(i)$ is $a \cdot w_{ij}$.

- Each unlabeled document is also connected to $k'NN_U(i)$, its k' nearest *unlabeled documents* (excluding itself). The weight between x_i and $x_j \in k'NN_U(i)$ is $b \cdot w_{ij}$. We also want $f(x_i)$ to be consistent with its similar unlabeled neighbors. We allow potentially different numbers of neighbors (k and k') and different weight coefficients (a and b) for unlabeled-labeled and unlabeled-unlabeled edges. These parameters are set by cross validation in experiments.

The last two kinds of edges are the key to semi-supervised learning: They connect unobserved nodes and force ratings to be smooth throughout the graph, as we discuss in the next section.

9.2 Applying the Harmonic Function

With the graph defined, we can apply any of the algorithms discussed in Section 2.5 to carry out semi-supervised learning. The basic idea is that our rating function $f(x)$ should be *smooth*

with respect to the graph. Let $L = 1 \ldots l$ and $U = l+1 \ldots n$ be labeled and unlabeled review indices, respectively. With the graph in Figure 9.1, the objective to minimize can be written as

$$\sum_{i \in L} M(f(x_i) - y_i)^2 + \sum_{i \in U} (f(x_i) - \hat{y}_i)^2 + \sum_{i \in U} \sum_{j \in kNN_L(i)} a w_{ij} (f(x_i) - f(x_j))^2 + \sum_{i \in U} \sum_{j \in k'NN_U(i)} b w_{ij} (f(x_i) - f(x_j))^2. \quad (9.1)$$

To understand the role of the parameters, we define $\alpha = ak + bk'$ and $\beta = \frac{b}{a}$ so that (9.1) can be written as

$$\sum_{i \in L} M(f(x_i) - y_i)^2 + \sum_{i \in U} \Big[(f(x_i) - \hat{y}_i)^2 + \frac{\alpha}{k + \beta k'} \Big(\sum_{j \in kNN_L(i)} w_{ij} (f(x_i) - f(x_j))^2 + \sum_{j \in k'NN_U(i)} \beta w_{ij} (f(x_i) - f(x_j))^2 \Big) \Big]. \quad (9.2)$$

Thus β controls the relative weight between labeled neighbors and unlabeled neighbors; α is roughly the relative weight given to semi-supervised (non-dongle) edges.

We now can minimize the objective using techniques based on the harmonic function closed-form solution (2.15). Defining an $n \times n$ matrix $\widehat{W}$,

$$\widehat{W}_{ij} = \begin{cases} 0, & i \in L \\ w_{ij}, & j \in kNN_L(i) \\ \beta w_{ij}, & j \in k'NN_U(i). \end{cases} \quad (9.3)$$

Let

$$W = \max(\widehat{W}, \widehat{W}^\top) \quad (9.4)$$

be the symmetrized matrix and D the corresponding diagonal degree. The graph Laplacian matrix is then $\Delta = D - W$. Let C be a diagonal dongle weight matrix with

$$C_{ii} = \begin{cases} M, & i \in L \\ 1, & i \in U \end{cases}. \quad (9.5)$$

Let

$$\mathbf{f} = (f(x_1), \ldots, f(x_n))^\top \quad (9.6)$$

$$\mathbf{y} = (y_1, \ldots, y_l, \hat{y}_{l+1}, \ldots, \hat{y}_n)^\top \quad . \quad (9.7)$$

We can rewrite (9.1) as

$$(\mathbf{f}-\mathbf{y})^\top C(\mathbf{f}-\mathbf{y}) + \frac{\alpha}{k+\beta k'}\mathbf{f}^\top \Delta \mathbf{f}. \tag{9.8}$$

This is a quadratic function in $\mathbf{f}$. Setting the gradient to zero, we find the minimum loss is obtained using

$$\mathbf{f} = \left(C + \frac{\alpha}{k+\beta k'}\Delta\right)^{-1} C\mathbf{y}. \tag{9.9}$$

Because C has strictly positive eigenvalues, the inverse is well defined.

Before moving on to experiments, we note an interesting connection to Pang and Lee's *metric labeling* method (2005). Consider a special case of our loss function (9.1) when $b = 0$ and $M \to \infty$. It is easy to show for labeled nodes $j \in L$, the optimal value is the given label: $f(x_j) = y_j$. Then the optimization problem decouples into a set of one-dimensional problems, one for each unlabeled node $i \in U$: $\mathcal{L}_{b=0,M\to\infty}(f(x_i)) =$

$$(f(x_i)-\hat{y}_i)^2 + \sum_{j\in kNN_L(i)} a w_{ij}(f(x_i)-y_j)^2. \tag{9.10}$$

The above problem is easy to solve. It corresponds exactly to the metric labeling method, except we use squared difference while Pang and Lee (2005) used absolute difference. Indeed in experiments comparing the two (not reported here), their differences are not statistically significant. From this perspective, our semi-supervised learning method is an extension with interacting terms among unlabeled data.

9.3 Experiments

We performed experiments using the movie review documents and accompanying 4-class ($C = \{0,1,2,3\}$) labels found in the "scale dataset v1.0"[1] developed and first used by Pang and Lee (2005). We chose 4-class instead of 3-class labeling because it is harder. The dataset is divided into four author-specific corpora, containing 1770, 902, 1307, and 1027 documents. We ran experiments individually for each author. Each document is represented as a $\{0,1\}$ word presence indicator vector, normalized to sum to 1.

[1] Available at `http://www.cs.cornell.edu/people/pabo/movie-review-data/`.

We systematically vary labeled set size $|L| \in \{0.9n, 800, 400, 200, 100, 50, 25, 12, 6\}$ to observe the effect of semi-supervised learning. $|L| = 0.9n$ is included to match 10-fold cross validation (Pang and Lee, 2005). For each $|L|$ we run 20 trials where we randomly split the corpus into labeled and test (unlabeled) sets. The same random splits are used for all methods, allowing paired t-tests for statistical significance. All results we report are average test set accuracy.

We compare our graph-based semi-supervised method with two previously studied methods: regression and metric labeling (Pang and Lee, 2005).

9.3.1 Regression

We ran linear ϵ-insensitive support vector regression using SVM^{light} (Joachims, 1999a) with all default parameters. The continuous prediction on a test document is discretized for classification. Regression results are reported under the heading "reg." Note this method does not use unlabeled data for training.

9.3.2 Metric labeling

We ran Pang and Lee's metric labeling method (2005), using SVM regression as the initial label preference function. The method requires an item-similarity function, which is equivalent to our similarity measure w_{ij}. Among others, we experimented with PSP-based similarity. Metric labeling results with this measure are reported under "reg+PSP." Note this method does not use unlabeled data for training either.

PSP$_i$ is defined as the percentage of positive sentences in review x_i (Pang and Lee, 2005). The similarity between reviews x_i, x_j is the cosine of the angle between the vectors $(\text{PSP}_i, 1-\text{PSP}_i)$ and $(\text{PSP}_j, 1 - \text{PSP}_j)$. Positive sentences are identified using a binary classifier trained on a separate "snippet dataset" located at the same URL as above. The snippet dataset contains 10662 short quotations taken from movie reviews appearing on the `rottentomatoes.com` Web site. Each snippet is labeled positive or negative based on the rating of the originating review. Pang and Lee (2005) trained a Naïve Bayes classifier. They show that PSP is a (noisy) measure for comparing reviews—reviews with low ratings tend to receive low PSP scores, and those with higher ratings

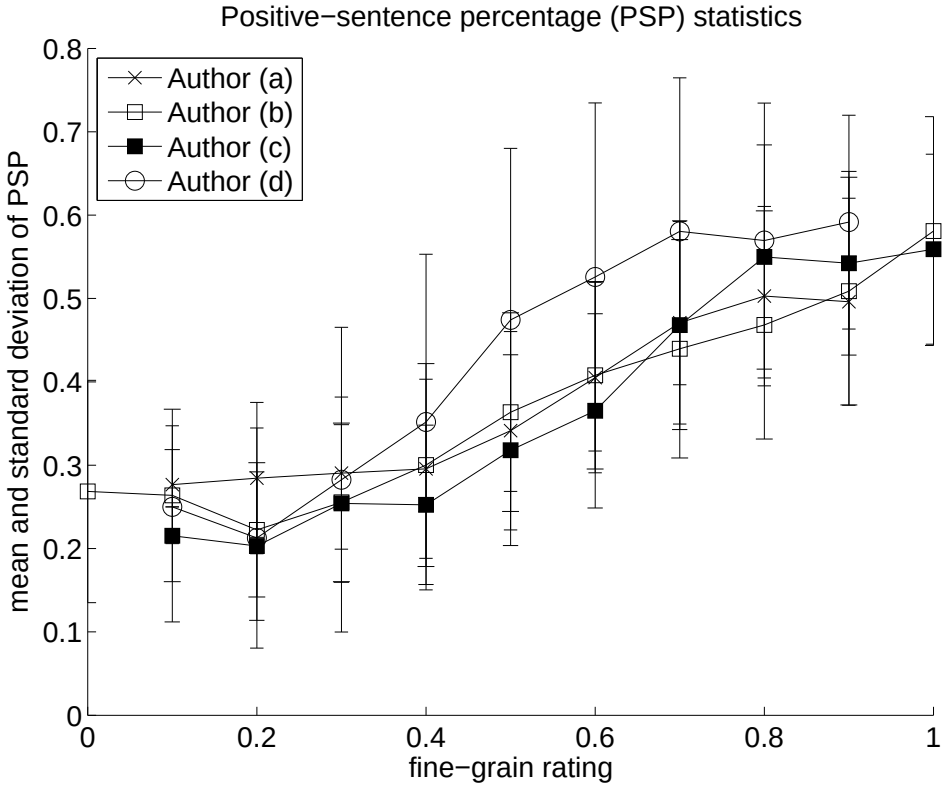


Figure 9.2: Positive Sentence Percentage (PSP) for reviews expressing each fine-grain rating. We identified positive sentences using SVM instead of Naïve Bayes, but the trend is qualitatively the same as in previous work (Pang and Lee, 2005).

tend to get high PSP scores. Thus, two reviews with a high PSP-based similarity are expected to have similar ratings. For our experiments we derived PSP measurements in a similar manner, but using a linear SVM classifier. We observed the same relationship between PSP and ratings (Figure 9.2).

The metric labeling method has parameters (the equivalent of k, α in our model). Pang and Lee (2005) tuned them on a per-author basis using cross validation but did not report the optimal parameters. We were interested in finding a single set of parameters for use with all authors. In addition, since we varied labeled set size, it is convenient to tune $c = k/|L|$, the fraction of labeled reviews used as neighbors, instead of k. We then used the same c, α for all authors at all labeled set sizes in experiments involving PSP. In an attempt to reproduce the findings of Pang and Lee

Author	reg	reg+PSP (shared)	reg+PSP (specific)
(a)	**0.592**	**0.592**	**0.592** ($c = 0.05$, $\alpha = 0.01$)
(b)	**0.501**	**0.498**	**0.496** ($c = 0.05$, $\alpha = 3.50$)
(c)	**0.592**	0.589	**0.593** ($c = 0.15$, $\alpha = 1.50$)
(d)	**0.496**	**0.498**	**0.500** ($c = 0.05$, $\alpha = 3.00$)

Table 9.1: Accuracy using shared ($c = 0.2$, $\alpha = 1.5$) versus author-specific parameters, with $|L| = 0.9n$.

(2005), we tuned c, α with cross validation. Tuning ranges are $c \in \{0.05, 0.1, 0.15, 0.2, 0.25, 0.3\}$ and $\alpha \in \{0.01, 0.1, 0.5, 1.0, 1.5, 2.0, 2.5, 3.0, 3.5, 4.0, 5.0\}$. The optimal parameters we found are $c = 0.2$ and $\alpha = 1.5$. (In section 9.3.4, we discuss an alternative similarity measure, for which we retuned these parameters.)

Note that we tuned a single set of shared parameters for all authors, whereas Pang and Lee (2005) tuned k and α on a per-author basis. To demonstrate that our implementation of metric labeling produces comparable results, we also determined the optimal author-specific parameters. Table 9.1 shows the accuracy obtained over 20 trials with $|L| = 0.9n$ for each author, using SVM regression, reg+PSP using shared c, α parameters, and reg+PSP using author-specific c, α parameters (listed in parentheses). The best result in each row of the table is highlighted in bold. We also show in bold any results that cannot be distinguished from the best result using a paired t-test at the 0.05 level.

Pang and Lee (2005) found that their metric labeling method, when applied to the 4-class data we are using, was not statistically better than regression, though they observed some improvement for authors (c) and (d). Using author-specific parameters, we obtained the same qualitative result, but the improvement for (c) and (d) appears even less significant in our results. Possible explanations for this difference are the fact that we derived our PSP measurements using an SVM classifier instead of an NB classifier, and that we did not use the same range of parameters for tuning. The optimal shared parameters produced almost the same results as the optimal author-specific parameters, and were used in subsequent experiments.

9.3.3 Semi-Supervised Learning

We used the same PSP-based similarity measure and the same shared parameters $c = 0.2, \alpha = 1.5$ from our metric labeling experiments to perform graph-based semi-supervised learning. The results are reported as "SSL+PSP." SSL has three additional parameters k', β, and M. Again we tuned k', β with cross validation. We considered the following tuning ranges: $k' \in \{2, 3, 5, 10, 20\}$ and $\beta \in \{0.001, 0.01, 0.1, 1.0, 10.0\}$. The optimal parameters are $k' = 5$ and $\beta = 1.0$. These were used for all authors and for all labeled set sizes. Note that unlike $k = c|L|$, which decreases as the labeled set size decreases, we let k' remain fixed for all $|L|$. We set M arbitrarily to a large number 10^8 to ensure that the ratings of labeled reviews are respected.

9.3.4 Alternate Similarity Measures

In addition to using PSP as a similarity measure between reviews, we investigated several alternative similarity measures based on the cosine of word vectors. Among these options were the cosine between the word vectors used to train the SVM regressor, and the cosine between word vectors containing only words with high (top 1000 or top 5000) mutual information values. The mutual information is computed with respect to the positive and negative classes in the 10662-document snippet dataset. Finally, we experimented with using as a similarity measure the cosine between word vectors containing all words, each weighted by its mutual information. We found this measure to be the best among the options tested in pilot trial runs using the metric labeling algorithm. Specifically, we scaled the mutual information values such that the maximum value was one. Then, we used these values as weights for the corresponding words in the word vectors. For words in the movie review dataset that did not appear in the snippet dataset, we used a default weight of zero (i.e., we excluded them. We experimented with setting the default weight to one, but found this led to inferior performance.)

We repeated the experiments described in sections 9.3.2 and 9.3.3 with the only difference being that we used the mutual-information weighted word vector similarity instead of PSP whenever a similarity measure was required. We repeated the tuning procedures described in the previous

sections. Using this new similarity measure led to the optimal parameters $c = 0.1$, $\alpha = 1.5$, $k' = 5$, and $\beta = 10.0$. The results are reported under "reg+WV" and "SSL+WV," respectively.

9.3.5 Results

We tested the five algorithms for all four authors using each of the nine labeled set sizes. The results are presented in table 9.2. Each entry in the table represents the average accuracy across 20 trials for an author, a labeled set size, and an algorithm. The best result in each row is highlighted in bold. Any results in the same row that cannot be distinguished from the best result using a paired t-test at the 0.05 level are also bold.

The results indicate that the graph-based semi-supervised learning algorithm based on PSP similarity (SSL+PSP) achieved better performance than all other methods in all four author corpora when only 200, 100, 50 or 25 labeled documents were available. In 15 out of these 16 learning scenarios, the unlabeled set accuracy by the SSL+PSP algorithm was significantly higher than all other methods. While accuracy generally degraded as we trained on less labeled data, the decrease for the SSL approach was less severe through the mid-range labeled set sizes. SSL+PSP remains among the best methods with only 12 or 6 labeled examples.

Note that, as discussed previously, an SSL algorithm like this one may be sensitive to graph weights (i.e., the similarity measure used to form the graph). In the experiments where we used mutual-information weighted word vector similarity (reg+WV and SSL+WV), we notice that reg+WV remained on par with reg+PSP at high labeled set sizes, whereas SSL+WV appears significantly worse in most of these cases. It is clear that PSP is the more reliable similarity measure. SSL exploits similarity more than metric labeling (i.e., SSL's graph is denser), so it is not surprising that SSL's accuracy would suffer more with an inferior similarity measure.

Interestingly, our SSL approach fared less well with large labeled set sizes. We believe this is due to two factors: a) the baseline SVM regressor trained on a large labeled set can achieve fairly high accuracy for this difficult task without considering pairwise relationships between examples; b) PSP similarity is not accurate enough. Gain in variance reduction achieved by the SSL graph is offset by its bias when labeled data is abundant.

			PSP		word vector	
	$\|L\|$	regression	reg+PSP	SSL+PSP	reg+WV	SSL+WV
Author (a)	1593	**0.592**	**0.592**	0.546	**0.592**	0.544
	800	**0.553**	**0.554**	0.534	**0.553**	0.517
	400	**0.522**	**0.525**	**0.526**	**0.522**	0.497
	200	0.494	0.498	**0.521**	0.494	0.472
	100	0.463	0.477	**0.511**	0.462	0.450
	50	0.439	0.458	**0.499**	0.438	0.429
	25	0.408	0.421	**0.465**	0.400	0.404
	12	0.411	**0.445**	**0.451**	0.341	0.410
	6	**0.391**	0.360	**0.404**	0.336	**0.390**
Author (b)	811	0.501	0.498	0.481	**0.503**	0.473
	800	0.501	0.497	0.478	**0.503**	0.474
	400	**0.471**	**0.471**	**0.465**	**0.471**	0.450
	200	**0.447**	**0.449**	**0.452**	**0.447**	0.429
	100	0.415	0.423	**0.443**	0.415	0.397
	50	0.388	0.396	**0.434**	0.387	0.376
	25	0.373	0.380	**0.418**	0.364	0.367
	12	0.352	**0.388**	**0.396**	0.318	0.351
	6	**0.353**	**0.364**	**0.363**	0.308	**0.353**
Author (c)	1176	**0.592**	**0.589**	0.566	**0.594**	0.514
	800	0.579	**0.585**	0.559	0.579	0.509
	400	0.550	**0.556**	0.544	0.551	0.491
	200	0.513	0.519	**0.532**	0.513	0.479
	100	0.484	0.495	**0.521**	0.484	0.466
	50	0.462	0.476	**0.504**	0.461	0.456
	25	0.459	0.472	**0.484**	0.439	0.454
	12	0.434	0.456	**0.472**	0.351	0.433
	6	**0.412**	**0.423**	**0.443**	0.358	**0.413**
Author (d)	924	**0.496**	**0.498**	**0.495**	**0.499**	**0.490**
	800	0.500	**0.501**	**0.495**	**0.504**	0.483
	400	0.474	0.478	**0.486**	0.477	0.463
	200	0.459	0.459	**0.468**	0.459	0.445
	100	0.444	0.445	**0.460**	0.444	0.437
	50	0.429	0.431	**0.445**	0.429	0.428
	25	0.411	0.411	**0.425**	0.400	0.409
	12	**0.388**	**0.405**	**0.404**	0.330	**0.388**
	6	**0.375**	**0.366**	**0.389**	0.329	**0.370**

Table 9.2: 20-trial average unlabeled set accuracy for each author across different labeled set sizes and methods. In each row, we list in bold the best result and any results that cannot be distinguished from it with a paired t-test at the 0.05 level.

9.4 Conclusions

Companies, politicians, and organizations can benefit from being able to automatically assign accurate ratings to reviews of products and other items discussed online. Reducing the cost and manual labeling effort required to reach acceptable accuracy is therefore of great practical value. We demonstrated the benefit of using unlabeled data for the sentiment analysis task of rating inference. There are several directions to improve the work, though, including better text processing and similarity measures. From the perspective of semi-supervised learning, there are two main lines of future research: 1. Our method is transductive: new reviews must be added to the graph before they can be classified. An extension to the inductive learning setting is possible using the approach of, e.g., Sindhwani et al. (2005a). 2. Develop models for semi-supervised cross-domain sentiment analysis (e.g., train on movie reviews, but test on product reviews). In this setting, labeled data is only available in a source domain, and the goal is to use unlabeled (source and) target data to improve domain transfer.

Part IV

Conclusion

Chapter 10

Summary and Future Work

10.1 Key Contributions

This dissertation begins to solve several open problems in SSL, where the goal is to augment a little labeled data with large amounts of unlabeled data to improve classification or regression performance. Labeled data can be time-consuming and costly to obtain, often requiring annotators with particular expertise. Unlabeled data is all around us, though, in the form of Web pages, news articles, query logs, sound recordings, large digital photo collections, and so on. The key question in semi-supervised learning research is how to extract knowledge and practical value out of unlabeled resources in a wide range of learning environments.

We propose solutions to several of the large challenges in this area by introducing the setting of online SSL and efficient algorithms that can learn effectively within this regime. Table 10.1 summarizes the contributions in online SSL from Part II. Both Online Manifold Regularization (Chapter 3) and OASIS (Chapter 4) are able to achieve constant time and space complexity per time step through sparse approximations without significant degradations in learner accuracy. The two methods make very different assumptions about the underlying data, however. The first assumes a low-dimensional manifold or graph structure upon which nearby examples possess the same labels, and the second assumes the classes are separated by a low-density gap. The two online approaches also differ in their basic mode of learning. Whereas Online Manifold Regularization moves between point estimates of a kernel-based classifier via stochastic gradient descent, OASIS applies online Bayesian updates to iteratively refine an approximation of the posterior distribution

Ch.	Name	SSL Assumption	Classifier Type	Active?
3	Online Manifold Regularization	Manifold / Graph	Frequentist / Kernel	No
4	OASIS	Low-Density Gap / Cluster	Bayesian / Linear	Yes

Table 10.1: Summary of online semi-supervised learning contributions.

over the space of weight vectors parameterizing a linear classifier. Lastly, OASIS is able to use this posterior distribution to perform a principled form of active learning.

In Part III, the dissertation introduces several novel assumptions that extend the reach of SSL by incorporating new forms of weak side information and prior knowledge. These contributions are summarized in Table 10.2, where the third and fourth columns describe the key underlying assumption allowing unlabeled data to be of value in the learning process. Multi-manifold SSL (Chapter 5) relaxes the assumption underlying graph-based SSL methods and assumes that examples lying on the same underlying low-dimensional manifold have the same label. The low-rank assumption proposed in Chapter 6 essentially assumes that examples sharing the same small number of latent factors possess the same labels (for one or more tasks). As discussed in the chapter, though, our method technically assumes that the label-by-item and feature-by-item matrices are jointly low-rank. As a result, our method enables multi-label prediction, transduction, and missing data imputation to be addressed simultaneously. The work on dissimilarity-based SSL and kernel regression using order preferences in Chapters 7–8 introduces additional forms of regularization based on new kinds of side information. As demonstrated in these chapters, domain-specific heuristics can be applied to generate many dissimilarity relationships and order preferences. Despite potential noise in this side information, the resulting SSL techniques greatly improve performance over supervised learning. Finally, Chapter 9 focuses on the application of graph-based SSL to the sentiment analysis task of rating inference. The work shows that using a novel "sentiment graph," which encodes several assumptions specifically designed for this task, leads to effective performance with only a very small number of labels.

Ch.	Novel Assumption	Assumes if $\mathbf{x}$ _____	then y _____
5	Multi-manifold (classification)	similar & on same manifold	same label
6	Low-Rank (multi-label, missing data)	same latent factor loadings	same labels
7	Dissimilarity (classification)	dissimilar	different label
8	Order Preferences (regression)	arbitrary pairwise relation	specific target value ordering
9	Sentiment Graph (ordinal classification)	similar positive-sentence %	similar label

Table 10.2: Summary of new assumptions allowing unlabeled data to improve learning in various classification and regression settings.

10.2 Future Challenges for SSL

Despite the advances put forth in this body of work, some key issues still remain. We now briefly summarize a few of these future challenges:

- "safe" semi-supervised learning that is resilient to incorrect model assumptions,
- unifying multiple types of relations between labeled and unlabeled examples,
- non-topical text classification using limited supervision,
- domain adaptation using only unlabeled target-domain data.

10.2.1 "Safe" Semi-Supervised Learning

SSL methods are able to learn using unlabeled data through one of several critical model assumptions (cluster, manifold, and others), as discussed throughout this dissertation. As it is difficult to test which assumptions hold in practice, and performance may suffer if the wrong algorithm is chosen, "safe" semi-supervised learning algorithms that are guaranteed to perform at least as well as supervised learning have great potential. A practitioner should be able to exploit unlabeled data

without being an SSL expert. Our recent empirical study of SSL algorithms applied to many natural language processing tasks (Goldberg and Zhu, 2009) shows that, contrary to popular wisdom, k-fold cross validation with as few as 10 labeled examples can be used to achieve one form of safe SSL by simply choosing among different supervised and semi-supervised algorithms. It is still desirable to find a more elegant and theoretically justified form of safe SSL. Bayesian modeling may be able to provide a solution by maintaining a posterior distribution over classifiers of multiple types. The key challenge is to define an intelligent prior over assumptions and classifiers, as well as Bayesian formulations of the different types of SSL learners so that we can define a proper likelihood function.

SSL can also be made safer by developing robust graph-based methods and less restrictive assumptions that are more likely to hold in richer, complex datasets. For example, Chapter 5 discussed how to exploit local geometry to detect changes in dimensionality, orientation, and density in order to learn with data supported on multiple intersecting manifolds, which occurs in applications such as object tracking and handwritten character recognition. Another desired future outcome is the development of a technique that can perform automatic graph selection for graph-based SSL and tolerate the absence of any true underlying manifold structure. Posing this as an optimization problem, where some variables select among graphs or graph components, may be one way to solve this problem.

10.2.2 Unifying Multiple Types of Relations in Graph-Based SSL

This dissertation has presented several novel algorithms that exploit relations between labeled and unlabeled examples in various types of learning settings and real-world applications. Many other types of relations remain unexplored, which can be utilized in graph-based transductive learning (also known as collective classification). For example, imagine trying to predict category labels (e.g., market sector) or numeric values (e.g., income level) for people in a social network, where people may be associated with one another directly through friendship, or indirectly by common interests, geographic locality, or having clicked on the same advertisements, to name just

a few types of relations. Each link may indicate similarity, dissimilarity, or other assumed relationships between the target values. Deciding how to integrate these sources of information in a semi-supervised setting is of large practical value. Chapters 7 and 8 have considered the use of dissimilarity relationships and order preferences, respectively, while other work in graph-based SSL largely relies on basic similarity. A long term goal is to incorporate (and trade-off between) disparate types of relations in a unified framework capable of solving real data mining and social networking tasks.

10.2.3 Non-topical Text Classification with Limited Supervision

Semi-supervised learning is particularly applicable to problems in sentiment analysis and blog analytics, as opinion-bearing blog posts and other forms of community-generated online content rarely come with explicit annotations. Likewise, little accurately labeled data exists for query-intent classification: predicting the desired intent of short Web search queries. These problems of great relevance in both the public and private sectors often deal with subtle non-topical class definitions requiring customized SSL algorithms and the use external resources, such as massive logs of unlabeled data, to achieve sufficient performance. While this dissertation describes one such application: predicting opinions in movie reviews on a 1–4 star rating scale (Chapter 9), and other previous research explores the use of unlabeled resources for wish recognition (Goldberg and Zhu, 2009) and query-intent classification (Fuxman et al., 2009), one overarching future goal is to develop data-efficient methods to tackle other non-topical classification problems, such as predicting the quality, difficulty, or reading level of arbitrary natural language text.

10.2.4 Domain Adaptation Using Only Unlabeled Target-Domain Data

Finally, much work in semi-supervised learning, perhaps with minor modifications, can be applied to the challenging yet extremely important problem of domain adaptation (also known as transfer learning). In this situation, we have some labeled data in a particular source domain (e.g., movie reviews), but it is too difficult or costly to annotate data in a desired target domain (e.g., product reviews). The goal is to learn relationships between the domains in order to transfer

knowledge gleaned from the source labeled data to the target unlabeled data. Though transfer learning poses new challenges, such as mapping between different feature spaces, ideas from SSL and learning structure from unlabeled data are certainly relevant. One idea, inspired by space-time physics, for tackling this problem involves using a diffusion model or random walk process. The key insight is to simultaneously identify clusters of words or features within and between domain (multiple "universes"). Within each domain, the process could behave much like a classical Markov Chain random walk (e.g., PageRank), yet can also allow probability mass to spread from the source to the target universe through "wormholes" (akin to "pivot features" or auxiliary tasks in related work). Given this basic formulation and careful selection of transition probabilities within and between universes, we can learn translation probabilities between domains to apply a classifier learned in the source domain to examples from the target domain. Once all examples (labeled and unlabeled) are normalized to a standard vocabulary, existing semi-supervised learning techniques may be used to transductively classify the unlabeled target data or train an inductive classifier for future use.

10.3 Final Summary

The research in this dissertation advances the state-of-the-art in SSL by contributing to many active areas of research, including problem formulation, learning theory, algorithm development, and the application of these ideas to real-world problems in natural language processing, computer vision, bioinformatics, and other challenging domains. Given the cost and difficulty in obtaining large amounts of labeled data, it is becoming increasingly important to continue developing new SSL algorithms (as well as algorithms for related areas like transfer learning or active learning) that can handle real-world learning settings while mitigating the risks involved in making restrictive assumptions regarding the use of unlabeled data. The work in this dissertation constitutes a major step in this direction, and the future work proposed here may help to one day realize the full power of unlabeled data.

Bibliography

S. Abney. *Semisupervised Learning for Computational Linguistics*. Chapman & Hall/CRC, 2007.

Y. Altun, D. McAllester, and M. Belkin. Maximum margin semi-supervised learning for structured variables. In *Advances in Neural Information Processing Systems (NIPS)*. MIT Press, Cambridge, MA, 2005.

M. Amini, F. Laviolette, and N. Usunier. A transductive bound for the voted classifier with an application to semi-supervised learning. In D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, editors, *Advances in Neural Information Processing Systems (NIPS)*. MIT Press, Cambridge, MA, 2009.

R. Ando and T. Zhang. A framework for learning predictive structures from multiple tasks and unlabeled data. *Journal of Machine Learning Research*, 6:1817–1853, 2005.

A. Argyriou. Efficient approximation methods for harmonic semi-supervised learning. Master's thesis, University College London, 2004.

A. Argyriou, C. A. Micchelli, and M. Pontil. On spectral learning. *Journal of Machine Learning Research*, 11:935–953, 2010.

A. Azran. The rendezvous algorithm: Multiclass semi-supervised learning with Markov random walks. In *Proceedings of the 24th Annual International Conference on Machine Learning (ICML)*, 2007.

M.-F. Balcan and A. Blum. An augmented pac model for semi-supervised learning. In O. Chapelle, B. Schölkopf, and A. Zien, editors, *Semi-Supervised Learning*. MIT Press, 2006.

M.-F. Balcan and A. Blum. A PAC-style model for learning from labeled and unlabeled data. In *Proceedings of the Conference on Learning Theory (COLT)*, 2005.

M.-F. Balcan, A. Blum, P. P. Choi, J. Lafferty, B. Pantano, M. R. Rwebangira, and X. Zhu. Person identification in webcam images: An application of semi-supervised learning. In *ICML 2005 Workshop on Learning with Partially Classified Training Data*, 2005a.

M.-F. Balcan, A. Blum, and K. Yang. Co-training and expansion: Towards bridging theory and practice. In L. K. Saul, Y. Weiss, and L. Bottou, editors, *Advances in Neural Information Processing Systems (NIPS)*. MIT Press, Cambridge, MA, 2005b.

S. Baluja. Probabilistic modeling for face orientation discrimination: Learning from labeled and unlabeled data. In M. J. Kearns, S. A. Solla, and D. A. Cohn, editors, *Advances in Neural Information Processing Systems (NIPS)*. MIT Press, Cambridge, MA, 1998.

S. Basu, M. Bilenko, A. Banerjee, and R. J. Mooney. Probabilistic semi-supervised clustering with constraints. In O. Chapelle, B. Schölkopf, and A. Zien, editors, *Semi-Supervised Learning*, pages 71–98. MIT Press, 2006.

S. Basu, I. Davidson, and K. Wagstaff, editors. *Constrained Clustering: Advances in Algorithms, Theory, and Applications*. Chapman & Hall/CRC Press, 2008.

M. Belkin, P. Niyogi, and V. Sindhwani. Manifold regularization: A geometric framework for learning from labeled and unlabeled examples. *Journal of Machine Learning Research*, 7:2399–2434, November 2006.

S. Ben-David, T. Lu, and D. Pal. Does unlabeled data provably help? worst-case analysis of the sample complexity of semi-supervised learning. In *Proceedings of the Conference on Learning Theory (COLT)*, 2008.

K. Bennett and A. Demiriz. Semi-supervised support vector machines. In *Advances in Neural Information Processing Systems (NIPS)*. MIT Press, Cambridge, MA, 1999.

M. Bernstein, V. de Silva, J. Langford, and J. Tenenbaum. Graph approximations to geodesics on embedded manifolds. Technical report, Stanford, 2000.

A. Beygelzimer, S. Dasgupta, and J. Langford. Importance weighted active learning. In *Proceedings of the 26th Annual International Conference on Machine Learning (ICML)*, 2009.

J. Bi, K. Bennett, M. Embrechts, C. Breneman, and M. Song. Dimensionality reduction via sparse support vector machines. *Journal of Machine Learning Research*, 3:1229–1243, 2003.

P. J. Bickel and B. Li. Local polynomial regression on unknown manifolds. In *IMS Lecture Notes Monograph Series, Complex Datasets and Inverse Problems: Tomography, Networks and Beyond*, volume 54, pages 177–186, 2007.

C. M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

A. Blum, J. Lafferty, M. Rwebangira, and R. Reddy. Semi-supervised learning using randomized mincuts. In *Proceedings of the 21st Annual International Conference on Machine Learning (ICML)*, 2004.

A. Blum and S. Chawla. Learning from labeled and unlabeled data using graph mincuts. In *Proceedings of the 18th Annual International Conference on Machine Learning (ICML)*, 2001.

A. Blum and T. Mitchell. Combining labeled and unlabeled data with co-training. In *Proceedings of the Workshop on Computational Learning Theory (COLT)*, 1998.

O. Bousquet, O. Chapelle, and M. Hein. Measure based regularization. In *Advances in Neural Information Processing Systems (NIPS)*. MIT Press, Cambridge, MA, 2004.

P. Bradley and O. Mangasarian. Feature selection via concave minimization and support vector machines. In *Proceedings of the 15th Annual International Conference on Machine Learning (ICML)*, California, 1998.

P. Bradley, K. Bennett, and A. Demiriz. Constrained k-means clustering. Technical Report MSR-TR-2000-65, Microsoft Research, 2000.

U. Brefeld and T. Scheffer. Semi-supervised learning for structured output variables. In *Proceedings of the 23rd Annual International Conference on Machine Learning (ICML)*, Pittsburgh, USA, 2006.

U. Brefeld, C. Büscher, and T. Scheffer. Multiview discriminative sequential learning. In *European Conference on Machine Learning (ECML)*, 2005.

U. Brefeld, T. Gaertner, T. Scheffer, and S. Wrobel. Efficient co-regularized least squares regression. In *Proceedings of the 23rd Annual International Conference on Machine Learning (ICML)*, Pittsburgh, USA, 2006.

C. Burges, T. Shaked, E. Renshaw, A. Lazier, M. Deeds, N. Hamilton, and G. Hullender. Learning to rank using gradient descent. In *Proceedings of the 22nd Annual International Conference on Machine Learning (ICML)*, 2005.

C. J. Burges and J. C. Platt. Semi-supervised learning with conditional harmonic mixing. In O. Chapelle, B. Schölkopf, and A. Zien, editors, *Semi-Supervised Learning*. MIT Press, Cambridge, MA, 2005.

C. Callison-Burch, D. Talbot, and M. Osborne. Statistical machine translation with word- and sentence-aligned parallel corpora. In *Proceedings of the ACL*, 2004.

E. J. Candès and B. Recht. Exact matrix completion via convex optimization. *Foundations of Computational Mathematics*, 9:717–772, 2009.

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

M. A. Carreira-Perpinan and R. S. Zemel. Proximity graphs for clustering and manifold learning. In L. K. Saul, Y. Weiss, and L. Bottou, editors, *Advances in Neural Information Processing Systems (NIPS)*. MIT Press, Cambridge, MA, 2005.

V. Castelli and T. Cover. The exponential value of labeled samples. *Pattern Recognition Letters*, 16(1):105–111, 1995.

G. Cavallanti, N. Cesa-Bianchi, and C. Gentile. Tracking the best hyperplane with a simple budget perceptron. *Machine Learning*, 69(2-3):143–167, 2007.

N. Cesa-Bianchi, A. Conconi, and C. Gentile. On the generalization ability of on-line learning algorithms. *IEEE Transactions on Information Theory*, 50(9):2050–2057, 2004.

O. Chapelle and A. Zien. Semi-supervised classification by low density separation. In *Proceedings of the Tenth International Workshop on Artificial Intelligence and Statistics (AISTAT 2005)*, 2005.

O. Chapelle, J. Weston, and B. Schölkopf. Cluster kernels for semi-supervised learning. In *Advances in Neural Information Processing Systems (NIPS)*. MIT Press, Cambridge, MA, 2002.

O. Chapelle, M. Chi, and A. Zien. A continuation method for semi-supervised SVMs. In *Proceedings of the 23rd Annual International Conference on Machine Learning (ICML)*, Pittsburgh, USA, 2006a.

O. Chapelle, V. Sindhwani, and S. S. Keerthi. Branch and bound for semi-supervised support vector machines. In *Advances in Neural Information Processing Systems (NIPS)*. MIT Press, Cambridge, MA, 2006b.

O. Chapelle, A. Zien, and B. Schölkopf, editors. *Semi-supervised learning*. MIT Press, 2006c.

O. Chapelle, V. Sindhwani, and S. S. Keerthi. Optimization techniques for semi-supervised support vector machines. *Journal of Machine Learning Research*, 9(Feb):203–233, 2008.

N. V. Chawla and G. Karakoulas. Learning from labeled and unlabeled data: An empirical study across techniques and domains. *Journal of Artificial Intelligence Research*, 23:331–366, 2005.

G. Chen and G. Lerman. Spectral curvature clustering. In *International Journal of Computer Vision*, 2008.

K. Chen and S. Wang. Regularized boost for semi-supervised learning. In J. Platt, D. Koller, Y. Singer, and S. Roweis, editors, *Advances in Neural Information Processing Systems (NIPS)*. MIT Press, Cambridge, MA, 2008.

W. Chu and Z. Ghahramani. Gaussian processes for ordinal regression. Technical report, University College London, 2004.

W. Chu, V. Sindhwani, Z. Ghahramani, and S. S. Keerthi. Relational learning with Gaussian processes. In *Advances in Neural Information Processing Systems (NIPS)*. MIT Press, Cambridge, MA, 2006.

W. Chu and Z. Ghahramani. Gaussian processes for ordinal regression. *Journal of Machine Learning Research*, 6(July):1019–1041, 2005.

W. Chu and S. S. Keerthi. New approaches to support vector ordinal regression. In *Proceedings of the 22nd Annual International Conference on Machine Learning (ICML)*, pages 145–152, Bonn, Germany, 2005.

F. R. K. Chung. *Spectral graph theory, Regional Conference Series in Mathematics, No. 92*. American Mathematical Society, 1997.

M. Collins and Y. Singer. Unsupervised models for named entity classification. In *EMNLP/VLC-99*, 1999.

R. Collobert, J. Weston, and L. Bottou. Trading convexity for scalability. In *Proceedings of the 23rd Annual International Conference on Machine Learning (ICML)*, Pittsburgh, USA, 2006.

A. Corduneanu and T. Jaakkola. On information regularization. In *Nineteenth Conference on Uncertainty in Artificial Intelligence (UAI03)*, 2003.

A. Corduneanu and T. Jaakkola. Stable mixing of complete and incomplete information. Technical Report AIM-2001-030, MIT AI Memo, 2001.

A. Corduneanu and T. S. Jaakkola. Distributed information regularization on graphs. In L. K. Saul, Y. Weiss, and L. Bottou, editors, *Advances in Neural Information Processing Systems (NIPS)*. MIT Press, Cambridge, MA, 2005.

C. Cortes, M. Mohri, D. Pechyony, and A. Rastogi. Stability of transductive regression algorithms. In *Proceedings of the 25th Annual International Conference on Machine Learning (ICML)*, 2008.

C. Cortes and M. Mohri. On transductive regression. In *Advances in Neural Information Processing Systems (NIPS)*. MIT Press, Cambridge, MA, 2006.

F. Cozman, I. Cohen, and M. Cirelo. Semi-supervised learning of mixture models. In *Proceedings of the 20th Annual International Conference on Machine Learning (ICML)*, 2003.

K. Crammer and Y. Singer. On the algorithmic implementation of multiclass kernel-based vector machines. *Journal of Machine Learning Research*, 2:265–292, 2002.

K. Crammer, A. Kulesza, and M. Dredze. Adaptive regularization of weight vectors. In Y. Bengio, D. Schuurmans, J. Lafferty, C. K. I. Williams, and A. Culotta, editors, *Advances in Neural Information Processing Systems (NIPS)*, pages 414–422. MIT Press, Cambridge, MA, 2009.

M. Culp and G. Michailidis. An iterative algorithm for extending learners to a semisupervised setting. In *The 2007 Joint Statistical Meetings (JSM)*, 2007.

G. Dai and D. Yeung. Kernel selection for semi-supervised kernel machines. In *Proceedings of the 24th Annual International Conference on Machine Learning (ICML)*, 2007.

R. Dara, S. Kremer, and D. Stacey. Clsutering unlabeled data with SOMs improves classification of labeled real-world data. In *Proceedings of the World Congress on Computational Intelligence (WCCI)*, 2002.

S. Dasgupta and Y. Freund. Random projection trees and low dimensional manifolds. Technical Report CS2007-0890, University of California, San Diego, 2007.

S. Dasgupta, M. L. Littman, and D. McAllester. PAC generalization bounds for co-training. In *Advances in Neural Information Processing Systems (NIPS)*. MIT Press, Cambridge, MA, 2001.

T. De Bie and N. Cristianini. Semi-supervised learning using semi-definite programming. In O. Chapelle, B. Schoëlkopf, and A. Zien, editors, *Semi-supervised learning*. MIT Press, Cambridge-Massachussets, 2006.

T. De Bie and N. Cristianini. Convex methods for transduction. In S. Thrun, L. Saul, and B. Schölkopf, editors, *Advances in Neural Information Processing Systems (NIPS)*. MIT Press, Cambridge, MA, 2004.

V. R. de Sa. Learning classification with unlabeled data. In *Advances in Neural Information Processing Systems (NIPS)*. Morgan Kaufmann, 1993.

O. Dekel, C. Manning, and Y. Singer. Loglinear models for label-ranking. In *Advances in Neural Information Processing Systems (NIPS)*. MIT Press, Cambridge, MA, 2003.

O. Dekel, S. Shalev-Shwartz, and Y. Singer. The forgetron: A kernel-based perceptron on a fixed budget. In *Advances in Neural Information Processing Systems (NIPS)*. MIT Press, Cambridge, MA, 2005.

O. Delalleau, Y. Bengio, and N. L. Roux. Efficient non-parametric function induction in semi-supervised learning. In *Proceedings of the Tenth International Workshop on Artificial Intelligence and Statistics (AISTAT 2005)*, 2005.

A. Demirez and K. Bennett. Optimization approaches to semisupervised learning. In M. Ferris, O. Mangasarian, and J. Pang, editors, *Applications and Algorithms of Complementarity*. Kluwer Academic Publishers, Boston, 2000.

A. Demiriz, K. Bennett, and M. Embrechts. Semi-supervised clustering using genetic algorithms. *Proceedings of Artificial Neural Networks in Engineering*, November 1999.

A. Dempster, N. Laird, and D. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, Series B*, 1977.

F. Denis, R. Gilleron, and M. Tommasi. Text classification from positive and unlabeled examples. In *The 9th International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems(IPMU)*, 2002.

A. Doucet and A. M. Johansen. A tutorial on particle filtering and smoothing: Fifteen years later. In D. Crisan and B. Rozovsky, editors, *Handbook of Nonlinear Filtering*. Oxford University Press, 2009.

A. Doucet, N. De Freitas, and N. Gordon, editors. *Sequential Monte Carlo methods in practice*. Springer, 2001.

M. B. Eisen, P. T. Spellman, P. O. Brown, and D. Botstein. Cluster analysis and display of genome-wide expression patterns. *Proceedings of the National Academy of Sciences of the United States of America*, 95(25):14863–14868, December 1998.

R. El-Yaniv and L. Gerzon. Effective transductive learning via objective model selection. *Pattern Recognition Letters*, 26(13):2104–2115, 2005.

R. El-Yaniv, D. Pechyony, and V. Vapnik. Large margin vs. large volume in transductive learning. *Machine Learning*, 72(3):173–188, 2008.

A. Elisseeff and J. Weston. A kernel method for multi-labelled classification. In T. G. Dietterich, S. Becker, and Z. Ghahramani, editors, *Advances in Neural Information Processing Systems (NIPS)*, pages 681–687. MIT Press, Cambridge, MA, 2001.

C. Elkan and K. Noto. Learning classifiers from only positive and unlabeled data. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 213–220, 2008.

D. Elworthy. Does Baum-Welch re-estimation help taggers? In *Proceedings of the 4th Conference on Applied Natural Language Processing*, 1994.

J. D. Farquhar, D. R. Hardoon, H. Meng, J. Shawe-Taylor, and S. Szedmak. Two view learning: SVM-2K, theory and practice. In *Advances in Neural Information Processing Systems (NIPS)*. MIT Press, Cambridge, MA, 2006.

Y. Freund, S. Dasgupta, M. Kabra, and N. Verma. Learning the structure of manifolds using random projections. In *Advances in Neural Information Processing System (NIPS)*. MIT Press, Cambridge, MA, 2007.

A. Fujino, N. Ueda, and K. Saito. A hybrid generative/discriminative approach to semi-supervised classifier design. In *AAAI-05, The Twentieth National Conference on Artificial Intelligence*, 2005.

A. Fujino, N. Ueda, and K. Saito. Semisupervised learning for a hybrid generative/discriminative classifier based on the maximum entropy principle. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 30(3):424–437, 2008.

G. Fung and O. Mangasarian. Semi-supervised support vector machines for unlabeled data classification. Technical Report 99-05, Data Mining Institute, University of Wisconsin Madison, October 1999.

A. Fuxman, A. Kannan, A. B. Goldberg, R. Agrawal, P. Tsaparas, and J. Shafer. Improving classification accuracy using automatically extracted training data. In *15th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD)*, June 2009.

J. Garcke and M. Griebel. Semi-supervised learning with sparse grids. In *Proc. of the 22nd ICML Workshop on Learning with Partially Classified Training Data*, Bonn, Germany, August 2005.

A. Gelman, A. Jakulin, M. G. Pittau, and Y.-S. Su. A weakly informative default prior distribution for logistic and other regression models. *Annals of Applied Statistics*, 2(4):1360–1383., 2008.

G. Getz, N. Shental, and E. Domany. Semi-supervised learning – a statistical physics approach. In *Proc. of the 22nd ICML Workshop on Learning with Partially Classified Training Data*, Bonn, Germany, August 2005.

Z. Ghahramani and M. I. Jordan. Supervised learning from incomplete data via an EM approach. In *Advances in Neural Information Processing Systems (NIPS)*, pages 120–127. Morgan Kaufmann, 1994.

W. R. Gilks and C. Berzuini. Following a moving target—Monte Carlo inference for dynamic Bayesian models. *Journal Of The Royal Statistical Society Series B*, 63(1):127–146, 2001.

J. Godfrey, E. Holliman, and J. McDaniel. Switchboard: Telephone speech corpus for research and development. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, March 1992.

A. B. Goldberg and X. Zhu. Keepin' it real: Semi-supervised learning with realistic tuning. In *NAACL HLT 2009 Workshop on Semi-supervised Learning for Natural Language Processing*, 2009.

A. B. Goldberg and X. Zhu. Seeing stars when there aren't many stars: Graph-based semi-supervised learning for sentiment categorization. In *HLT-NAACL 2006 Workshop on Textgraphs: Graph-based Algorithms for Natural Language Processing*, New York, NY, 2006.

A. B. Goldberg, X. Zhu, and S. Wright. Dissimilarity in graph-based semi-supervised classification. In *Eleventh International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2007.

A. B. Goldberg, M. Li, and X. Zhu. Online manifold regularization: A new learning setting and empirical study. In *The European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML PKDD)*, 2008.

A. B. Goldberg, X. Zhu, A. Singh, Z. Xu, and R. Nowak. Multi-manifold semi-supervised learning. In *Twelfth International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2009.

S. Goldman and Y. Zhou. Enhancing supervised learning with unlabeled data. In *Proceedings of the 17th Annual International Conference on Machine Learning (ICML)*, pages 327–334. Morgan Kaufmann, San Francisco, CA, 2000.

R. Gomes, M. Welling, and P. Perona. Incremental learning of nonparametric bayesian mixture models. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE Computer Society, 2008.

H. Grabner, C. Leistner, and H. Bischof. Semi-supervised on-line boosting for robust tracking. In *Proceedings of the 10th European Conference on Computer Vision (ECCV)*, pages 234–247, Berlin, Heidelberg, 2008. Springer-Verlag.

L. Grady and G. Funka-Lea. Multi-label image segmentation for medical applications based on graph-theoretic electrical potentials. In *European Conference on Computer Vision (ECCV) 2004 workshop*, 2004.

Y. Grandvalet and Y. Bengio. Semi-supervised learning by entropy minimization. In L. K. Saul, Y. Weiss, and L. Bottou, editors, *Advances in Neural Information Processing Systems (NIPS)*. MIT Press, Cambridge, MA, 2005.

N. Grira, M. Crucianu, and N. Boujemaa. Unsupervised and semi-supervised clustering: A brief survey. In 'A Review of Machine Learning Techniques for Processing Multimedia Content', Report of the MUSCLE European Network of Excellence (FP6), 2004.

G. Haffari and A. Sarkar. Analysis of semi-supervised learning with the Yarowsky algorithm. In *23rd Conference on Uncertainty in Artificial Intelligence (UAI)*, 2007.

G. Haro, G. Randall, and G. Sapiro. Translated poisson mixture model for stratification learning. *International Journal of Computer Vision*, 80:358–374, 2008.

W. Hastings. Monte Carlo sampling methods using Markov chains and their applications. *Biometrika*, 57:97–109, 1970.

C. Hegde, M. Wakin, and R. Baraniuk. Random projections for manifold learning. In *Advances in Neural Information Processing Systems (NIPS)*. MIT Press, Cambridge, MA, 2007.

M. Hein and M. Maier. Manifold denoising. In *Advances in Neural Information Processing Systems (NIPS)*. MIT Press, Cambridge, MA, 2006.

M. Hein, J.-Y. Audibert, and U. von Luxburg. Graph Laplacians and their convergence on random neighborhood graphs. *Journal of Machine Learning Research*, 8(Jun):1325–1368, 2007.

R. Herbrich, K. Obermayer, and T. Graepel. Large margin rank boundaries for ordinal regression. In A. J. Smola, P. Bartlett, B. Schölkopf, and D. Schuurmans, editors, *Advances in Large Margin Classifiers*, pages 115–132. MIT Press, 2000.

M. Herbster, M. Pontil, and S. R. Galeano. Fast prediction on a tree. In D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, editors, *Advances in Neural Information Processing Systems (NIPS)*. MIT Press, Cambridge, MA, 2009.

D. Hsu, S. Kakade, J. Langford, and T. Zhang. Multi-label prediction via compressed sensing. In Y. Bengio, D. Schuurmans, J. Lafferty, C. K. I. Williams, and A. Culotta, editors, *Advances in Neural Information Processing Systems (NIPS)*. MIT Press, Cambridge, MA, 2009.

B. Huang and T. Jebara. Loopy belief propagation for bipartite maximum weight b-matching. In M. Meila and X. Shen, editors, *Proceedings of the Eleventh International Conference on Artificial Intelligence and Statistics March 21-24, 2007, San Juan, Puerto Rico*, volume Volume 2 of JMLR: W&CP, March 2007.

T. Jaakkola, M. Meila, and T. Jebara. Maximum entropy discrimination. *Neural Information Processing Systems, 12*, 12, 1999.

T. Jebara, R. Kondor, and A. Howard. Probability product kernels. *Journal of Machine Learning Research, Special Topic on Learning Theory*, 5:819–844, 2004.

T. Jebara, J. Wang, and S. Chang. Graph construction and b-matching for semi-supervised learning. In *Proceedings of the 26th Annual International Conference on Machine Learning (ICML)*, 2009.

S. Ji, L. Tang, S. Yu, and J. Ye. Extracting shared subspace for multi-label classification. In *KDD '08: Proceeding of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 381–389, New York, NY, USA, 2008. ACM.

X. Ji. *Graph partition problems with minimum size constraints*. PhD thesis, Rensselaer Polytechnic Institute, Dept. of Mathematics, 2004.

T. Joachims. Optimizing search engines using clickthrough data. In *Proceedings of KDD '02, the ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM Press, 2002.

T. Joachims. Transductive learning via spectral graph partitioning. In *Proceedings of the 20th Annual International Conference on Machine Learning (ICML)*, 2003.

T. Joachims. Making large-scale SVM learning practical. In B. Schölkopf, C. Burges, and A. Smola, editors, *Advances in Kernel Methods - Support Vector Learning*. MIT Press, 1999a.

T. Joachims. Transductive inference for text classification using support vector machines. In *Proceedings of the 16th Annual International Conference on Machine Learning (ICML)*, pages 200–209. Morgan Kaufmann, San Francisco, CA, 1999b.

R. Johnson and T. Zhang. On the effectiveness of laplacian normalization for graph semi-supervised learning. *Journal of Machine Learning Research*, 8(Jul):1489–1517, 2007a.

R. Johnson and T. Zhang. Two-view feature generation model for semi-supervised learning. In *Proceedings of the 24th Annual International Conference on Machine Learning (ICML)*, 2007b.

R. Jones. Learning to extract entities from labeled and unlabeled text. Technical Report CMU-LTI-05-191, Carnegie Mellon University, 2005. Doctoral Dissertation.

M. Kaariainen. Generalization error bounds using unlabeled data. In *Proceedings of the Conference on Learning Theory (COLT)*, 2005.

A. Kapoor, Y. Qi, H. Ahn, and R. Picard. Hyperparameter and kernel learning for graph based semi-supervised classification. In *Advances in Neural Information Processing Systems (NIPS)*. MIT Press, Cambridge, MA, 2005.

M. Karlen, J. Weston, A. Erkan, and R. Collobert. Large scale manifold transduction. In *Proceedings of the 25th Annual International Conference on Machine Learning (ICML)*, 2008.

C. Kemp, T. Griffiths, S. Stromsten, and J. Tenenbaum. Semi-supervised learning with trees. In *Advances in Neural Information Processing System (NIPS)*. MIT Press, Cambridge, MA, 2003.

G. Kimeldorf and G. Wahba. Some results on Tchebychean spline functions. *Journal of Mathematics Analysis and Applications*, 33:82–95, 1971.

J. Kivinen, A. J. Smola, and R. C. Williamson. Online learning with kernels. *IEEE Transactions on Signal Processing*, 52(8):2165–2176, 2004.

R. I. Kondor and J. Lafferty. Diffusion kernels on graphs and other discrete input spaces. In *Proceedings of the 19th Annual International Conference on Machine Learning (ICML)*, 2002.

B. Krishnapuram, D. Williams, Y. Xue, A. Hartemink, L. Carin, and M. Figueiredo. On semi-supervised classification. In L. K. Saul, Y. Weiss, and L. Bottou, editors, *Advances in Neural Information Processing Systems (NIPS)*. MIT Press, Cambridge, MA, 2005.

D. Kushnir, M. Galun, and A. Brandt. Fast multiscale clustering and manifold identification. *Pattern Recognition*, 39:1876–1891, 2006.

J. Lafferty and L. Wasserman. Statistical analysis of semi-supervised regression. In *Advances in Neural Information Processing Systems (NIPS)*. MIT Press, Cambridge, MA, 2007.

J. Lafferty, X. Zhu, and Y. Liu. Kernel conditional random fields: Representation and clique selection. In *Proceedings of the 21st Annual International Conference on Machine Learning (ICML)*, 2004.

N. D. Lawrence and M. I. Jordan. Semi-supervised learning via Gaussian processes. In L. K. Saul, Y. Weiss, and L. Bottou, editors, *Advances in Neural Information Processing Systems (NIPS)*. MIT Press, Cambridge, MA, 2005.

G. Lebanon. *Riemannian Geometry and Statistical Machine Learning*. PhD thesis, Carnegie Mellon University, 2005. CMU-LTI-05-189.

Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, November 1998.

C.-H. Lee, S. Wang, F. Jiao, D. Schuurmans, and R. Greiner. Learning to model spatial dependency: Semi-supervised discriminative random fields. In *Advances in Neural Information Processing Systems (NIPS)*. MIT Press, Cambridge, MA, 2006.

W. S. Lee and B. Liu. Learning with positive and unlabeled examples using weighted logistic regression. In *Proceedings of the 20th International Conference on Machine Learning (ICML)*, 2003.

Y. Lee, Y. Lin, and G. Wahba. Multicategory support vector machines, theory, and application to the classification of microarray data and satellite radiance data. *Journal of the American Statistical Association*, 99:67–81, 2004.

B. Leskes. The value of agreement, a new boosting algorithm. In *Proceedings of the Conference on Learning Theory (COLT)*, 2005.

A. Levin, D. Lischinski, and Y. Weiss. Colorization using optimization. In *ACM Transactions on Graphics*, 2004.

Y. Li and C. Guan. Joint feature re-extraction and classification using an iterative semi-supervised support vector machine algorithm. *Machine Learning*, 71(1):33–53, 2008.

Z. Li, J. Liu, and X. Tang. Pairwise constraint propagation by semidefinite programming for semi-supervised classification. In A. McCallum and S. Roweis, editors, *Proceedings of the 25th Annual International Conference on Machine Learning (ICML)*. Omnipress, 2008.

R. J. A. Little and D. B. Rubin. *Statistical Analysis with Missing Data*. Wiley-Interscience, 2nd edition, September 2002.

B. Liu, W. S. Lee, P. S. Yu, and X. Li. Partially supervised classification of text documents. In *Proceedings of the 19th Annual International Conference on Machine Learning (ICML)*, 2002.

D. C. Liu and J. Nocedal. On the limited memory BFGS method for large scale optimization. *Mathematical Programming*, 45:503–528, 1989.

Q. Liu, X. Liao, and L. Carin. Semi-supervised multitask learning. In J. Platt, D. Koller, Y. Singer, and S. Roweis, editors, *Advances in Neural Information Processing Systems (NIPS)*. MIT Press, Cambridge, MA, 2008.

N. Loeff, D. Forsyth, and D. Ramachandran. Manifoldboost: stagewise function approximation for fully-, semi- and un-supervised learning. In *Proceedings of the 25th Annual International Conference on Machine Learning (ICML)*, 2008.

Q. Lu and L. Getoor. Link-based classification using labeled and unlabeled data. In *ICML 2003 workshop on The Continuum from Labeled to Unlabeled Data in Machine Learning and Data Mining*, 2003.

S. Ma, D. Goldfarb, and L. Chen. Fixed point and Bregman iterative methods for matrix rank minimization. *Mathematical Programming Series A*, page to appear (published online September 23), 2009.

Y. Ma, H. Derksen, W. Hong, and J. Wright. Segmentation of multivariate mixed data via lossy coding and compression. *PAMI*, 29(9):1546–1562, 2007.

O. Madani, D. M. Pennock, and G. W. Flake. Co-validation: Using model disagreement to validate classification algorithms. In L. K. Saul, Y. Weiss, and L. Bottou, editors, *Advances in Neural Information Processing Systems (NIPS)*. MIT Press, Cambridge, MA, 2005.

B. Maeireizo, D. Litman, and R. Hwa. Co-training for predicting emotions with spoken dialogue data. In *The Companion Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL)*, 2004.

M. Mahdaviani and T. Choudhury. Fast and scalable training of semi-supervised CRFs with application to activity recognition. In J. Platt, D. Koller, Y. Singer, and S. Roweis, editors, *Advances in Neural Information Processing Systems (NIPS)*. MIT Press, Cambridge, MA, 2008.

M. Mahdaviani, N. de Freitas, B. Fraser, and F. Hamze. Fast computational methods for visually guided robots. In *The 2005 International Conference on Robotics and Automation (ICRA)*, 2005.

O. L. Mangasarian, J. W. Shavlik, and E. W. Wild. Knowledge-based kernel approximation. *Journal of Machine Learning Research*, 5:1127–1141, 2004.

O. Mangasarian. Generalized support vector machines. In A. J. Smola, P. Bartlett, B. Schölkopf, and D. Schuurmans, editors, *Advances in Large Margin Classifiers*, pages 135–146. MIT Press, 2000.

G. S. Mann and A. McCallum. Simple, robust, scalable semi-supervised learning via expectation regularization. In *Proceedings of the 24th Annual International Conference on Machine Learning (ICML)*, 2007.

C. D. Manning and H. Schütze. *Foundations of Statistical Natural Language Processing*. The MIT Press, Cambridge, Massachusetts, 1999.

D. McClosky, E. Charniak, and M. Johnson. Effective self-training for parsing. In *Human Language Technologies: The Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL-HLT)*, pages 152–159, 2006.

N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller. Equation of state calculations by fast computing machines. *Journal of Chemical Physics*, 21:1087–1092, 1953.

D. Miller and H. Uyar. A mixture of experts classifier with learning based on both labelled and unlabelled data. In *Advances in Neural Information Processing Systems (NIPS)*. MIT Press, Cambridge, MA, 1997.

T. Mitchell. The role of unlabeled data in supervised learning. In *Proceedings of the Sixth International Colloquium on Cognitive Science*, San Sebastian, Spain, 1999.

T. M. Mitchell. *Machine Learning*. McGraw-Hill, New York, 1997.

M. Mizra, J. Sommers, P. Barford, and X. Zhu. A machine learning approach to TCP throughput prediction. In *ACM SIGMETRICS*, 2007.

P. Mordohai and G. Medioni. Unsupervised dimensionality estimation and manifold learning in high-dimensional spaces by tensor voting. In *IJCAI*, 2005.

T. Mullen and R. Malouf. A preliminary investigation into sentiment analysis for informal political discourse. In *Proceedings of the AAAI Workshop on Analysis of Weblogs*, 2006.

R. M. Neal. Markov chain sampling methods for dirichlet process mixture models. *Journal of Computational and Graphical Statistics*, 9(2):249–265, 2000.

K. Nigam and R. Ghani. Analyzing the effectiveness and applicability of co-training. In *Ninth International Conference on Information and Knowledge Management*, pages 86–93, 2000.

K. Nigam, A. K. McCallum, S. Thrun, and T. Mitchell. Text classification from labeled and unlabeled documents using EM. *Machine Learning*, 39(2/3):103–134, 2000.

Z.-Y. Niu, D.-H. Ji, and C.-L. Tan. Word sense disambiguation using label propagation based semi-supervised learning. In *Proceedings of the ACL*, 2005.

P. Niyogi. Manifold regularization and semi-supervised learning: Some theoretical analyses. Technical Report TR-2008-01, CS Dept, U. of Chicago, 2008.

R. Nowak. Noisy generalized binary search. In Y. Bengio, D. Schuurmans, J. Lafferty, C. K. I. Williams, and A. Culotta, editors, *Advances in Neural Information Processing Systems (NIPS)*, pages 1366–1374. MIT Press, Cambridge, MA, 2009.

G. Obozinski, B. Taskar, and M. I. Jordan. Joint covariate selection and joint subspace selection for multiple classification problems. *Statistics and Computing*, 20(2):231–252, 2010.

D. Opitz and R. Maclin. Popular ensemble methods: An empirical study. *Journal of Artificial Intelligence Research*, 11:169–198, 1999.

B. Pang and L. Lee. *Opinion Mining and Sentiment Analysis*. Now Publishers Inc, July 2008.

B. Pang and L. Lee. A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. In *Proceedings of the Association for Computational Linguistics*, pages 271–278, 2004.

B. Pang and L. Lee. Seeing stars: exploiting class relationships for sentiment categorization with respect to rating scales. In *Proceedings of the Association for Computational Linguistics*, 2005.

T. P. Pham, H. T. Ng, and W. S. Lee. Word sense disambiguation with semi-supervised learning. In *AAAI-05, The Twentieth National Conference on Artificial Intelligence*, 2005.

M. Polito and P. Perona. Grouping and dimensionality reduction by locally linear embedding. In *Advances in Neural Information Processing Systems (NIPS)*. MIT Press, Cambridge, MA, 2002.

A. Rahimi and B. Recht. Random features for large-scale kernel machines. In J. Platt, D. Koller, Y. Singer, and S. Roweis, editors, *Advances in Neural Information Processing Systems (NIPS)*. MIT Press, Cambridge, MA, 2007.

P. Rai and H. Daume. Multi-label prediction via sparse infinite CCA. In Y. Bengio, D. Schuurmans, J. Lafferty, C. K. I. Williams, and A. Culotta, editors, *Advances in Neural Information Processing Systems (NIPS)*, pages 1518–1526. MIT Press, Cambridge, MA, 2009.

R. Raina, A. Battle, H. Lee, B. Packer, and A. Y. Ng. Self-taught learning: Transfer learning from unlabeled data. In *Proceedings of the 24th Annual International Conference on Machine Learning (ICML)*, 2007.

M. Ranzato and M. Szummer. Semi-supervised learning of compact document representations with deep networks. In *Proceedings of the 25th Annual International Conference on Machine Learning (ICML)*, 2008.

J. Ratsaby and S. Venkatesh. Learning from a mixture of labeled and unlabeled examples with parametric side information. *Proceedings of the Eighth Annual Conference on Computational Learning Theory*, pages 412–417, 1995.

P. Ravikumar and J. Lafferty. Quadratic programming relaxations for metric labeling and markov random field MAP estimation. In *Proceedings of the 23rd Annual International Conference on Machine Learning (ICML)*, Pittsburgh, USA, 2006.

G. Ridgeway and D. Madigan. A sequential Monte Carlo method for Bayesian analysis of massive datasets. *Journal of Data Mining and Knowledge Discovery*, 7(3):301–319, 2003.

P. Rigollet. Generalization error bounds in semi-supervised classification under the cluster assumption. *Journal of Machine Learning Research*, 8(Jul):1369–1392, 2007.

E. Riloff, J. Wiebe, and T. Wilson. Learning subjective nouns using extraction pattern bootstrapping. In *Proceedings of the Seventh Conference on Natural Language Learning (CoNLL-2003)*, 2003.

C. Rosenberg, M. Hebert, and H. Schneiderman. Semi-supervised self-training of object detection models. In *Seventh IEEE Workshop on Applications of Computer Vision*, January 2005.

S. Rosset, J. Zhu, H. Zou, and T. Hastie. A method for inferring label sampling mechanisms in semi-supervised learning. In L. K. Saul, Y. Weiss, and L. Bottou, editors, *Advances in Neural Information Processing Systems (NIPS)*. MIT Press, Cambridge, MA, 2005.

B. Schölkopf and A. J. Smola. *Learning with Kernels*. MIT Press, 2002.

B. Schölkopf, R. Herbrich, and A. J. Smola. A generalized representer theorem. In *Proceedings of the Fourteenth Annual Conference on Computational Learning Theory*, 2001.

D. Schuurmans and F. Southey. Metric-based methods for adaptive model selection and regularization. *Machine Learning, Special Issue on New Methods for Model Selection and Model Combination*, 48:51–84, 2001.

M. Seeger. Learning with labeled and unlabeled data. Technical report, University of Edinburgh, 2001.

B. Settles. Active learning literature survey. Computer Sciences Technical Report 1648, University of Wisconsin–Madison, 2009.

B. Shahshahani and D. Landgrebe. The effect of unlabeled samples in reducing the small sample size problem and mitigating the Hughes phenomenon. *IEEE Trans. On Geoscience and Remote Sensing*, 32(5):1087–1095, September 1994.

V. Sindhwani and D. Rosenberg. An rkhs for multi-view learning and manifold co-regularization. In *Proceedings of the 25th Annual International Conference on Machine Learning (ICML)*, 2008.

V. Sindhwani and S. S. Keerthi. Large scale semisupervised linear SVMs. In *SIGIR 2006*, 2006.

V. Sindhwani, P. Niyogi, and M. Belkin. Beyond the point cloud: from transductive to semi-supervised learning. In *Proceedings of the 22nd Annual International Conference on Machine Learning (ICML)*, 2005a.

V. Sindhwani, P. Niyogi, and M. Belkin. A co-regularized approach to semi-supervised learning with multiple views. In *Proc. of the 22nd ICML Workshop on Learning with Multiple Views*, August 2005b.

V. Sindhwani, P. Niyogi, M. Belkin, and S. Keerthi. Linear manifold regularization for large scale semi-supervised learning. In *Proc. of the 22nd ICML Workshop on Learning with Partially Classified Training Data*, August 2005c.

V. Sindhwani, S. Keerthi, and O. Chapelle. Deterministic annealing for semi-supervised kernel machines. In *Proceedings of the 23rd Annual International Conference on Machine Learning (ICML)*, Pittsburgh, USA, 2006.

V. Sindhwani, J. Hu, and A. Mojsilovic. Regularized co-clustering with dual supervision. In D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, editors, *Advances in Neural Information Processing Systems (NIPS)*. MIT Press, Cambridge, MA, 2009.

A. Singh, R. Nowak, and X. Zhu. Unlabeled data: Now it helps, now it doesn't. In D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, editors, *Advances in Neural Information Processing Systems (NIPS)*, pages 1513–1520. MIT Press, Cambridge, MA, 2008.

K. Sinha and M. Belkin. The value of labeled and unlabeled examples when the model is imperfect. In J. Platt, D. Koller, Y. Singer, and S. Roweis, editors, *Advances in Neural Information Processing Systems (NIPS)*. MIT Press, Cambridge, MA, 2008.

A. Smola and R. Kondor. Kernels and regularization on graphs. In *Proceedings of the Conference on Learning Theory (COLT)*, 2003.

A. Smola and B. Schölkopf. A tutorial on support vector regression. *Statistics and Computing*, 14: 199–222, 2004.

N. Sokolovska, O. Cappé, and F. Yvon. The asymptotics of semi-supervised learning in discriminative probabilistic models. In *Proceedings of the 25th Annual International Conference on Machine Learning (ICML)*, 2008.

N. Srebro and A. Shraibman. Rank, trace-norm and max-norm. In *Proceedings of the 18th Annual Conference on Learning Theory*, pages 545–560. Springer-Verlag, 2005.

I. Sutskever. A simpler unified analysis of budget perceptrons. In *Proceedings of the 26th Annual International Conference on Machine Learning (ICML)*, 2009.

A. D. Szlam, M. Maggioni, and R. R. Coifman. Regularization on graphs with function-adapted diffusion processes. *Journal of Machine Learning Research*, 9(Aug):1711–1739, 2008.

M. Szummer and T. Jaakkola. Partially labeled classification with Markov random walks. In *Advances in Neural Information Processing Systems (NIPS)*, volume 14, Cambridge, MA, 2001. MIT Press.

M. Szummer and T. Jaakkola. Information regularization with partially labeled data. In *Advances in Neural Information Processing Systems (NIPS)*, volume 15. MIT Press, Cambridge, MA, 2002.

F. Tang, S. Brennan, Q. Zhao, and H. Tao. Co-tracking using semi-supervised support vector machines. In *IEEE 11th International Conference on Computer Vision (ICCV)*, pages 1–8, 2007.

B. Taskar, C. Guestrin, and D. Koller. Max-margin Markov networks. In *Advances in Neural Information Processing Systems (NIPS)*. MIT Press, Cambridge, MA, 2003.

W. Tong and R. Jin. Semi-supervised learning by mixed label propagation. In *Proceedings of the Twenty-Second AAAI Conference on Artificial Intelligence (AAAI)*, 2007.

K. Trohidis, G. Tsoumakas, G. Kalliris, and I. Vlahavas. Multilabel classification of music into emotions. In *Proc. 9th International Conference on Music Information Retrieval (ISMIR 2008), Philadelphia, PA, USA, 2008*, 2008.

R. Tron and R. Vidal. A benchmark for the comparison of 3-d motion segmentation algorithms. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2007.

I. Tsang and J. Kwok. Large-scale sparsified manifold regularization. In *Advances in Neural Information Processing Systems (NIPS)*. MIT Press, Cambridge, MA, 2006.

I. W. Tsang, J. T. Kwok, and P.-M. Cheung. Core vector machines: Fast svm training on very large data sets. *Journal of Machine Learning Research*, 6:363–392, 2005.

I. Tsochantaridis, T. Joachims, T. Hofmann, and Y. Altun. Large margin methods for structured and interdependent output variables. *Journal of Machine Learning Research*, 6:1453–1484, 2005.

G. Tsoumakas, I. Katakis, and I. Vlahavas. Mining multi-label data. In *Data Mining and Knowledge Discovery Handbook*. Springer, 2nd edition, 2010.

A. B. Tsybakov. *Introduction a l'estimation non-parametrique*. Springer, Berlin Heidelberg, 2004.

J. Van Gael and X. Zhu. Correlation clustering for crosslingual link detection. In *International Joint Conference on Artificial Intelligence (IJCAI)*, 2007.

V. Vapnik. *Statistical Learning Theory*. Wiley-Interscience, 1998.

R. Vidal, Y. Ma, and S. Sastry. *Generalized Principal Component Analysis (GPCA)*. Springer Verlag, 2008.

P. Vincent and Y. Bengio. Kernel matching pursuit. *Machine Learning*, 48(1-3):165–187, 2002.

U. von Luxburg, M. Belkin, and O. Bousquet. Consistency of spectral clustering. Technical Report TR-134, Max Planck Institute for Biological Cybernetics, 2004.

U. von Luxburg. A tutorial on spectral clustering. *Statistics and Computing*, 17(4):395–416, 2007.

K. Wagstaff, C. Cardie, S. Rogers, and S. Schrödl. Constrained k-means clustering with background knowledge. In *Proceedings of the 18th Annual International Conference on Machine Learning (ICML)*, page 577, 2001.

M. Wainwright, T. Jaakkola, and A. Willsky. MAP estimation via agreement on (hyper)trees: Message passing and linear-programming approaches. *IEEE Transactions on Information Theory*, 51:3697–3717, 2005.

F. Wang and C. Zhang. Label propagation through linear neighborhoods. In *Proceedings of the 23rd Annual International Conference on Machine Learning (ICML)*, Pittsburgh, USA, 2006.

H. Wang, S. Yan, T. Huang, J. Liu, and X. Tang. Transductive regression piloted by inter-manifold relations. In *Proceedings of the 24th Annual International Conference on Machine Learning (ICML)*, 2007.

J. Wang, T. Jebara, and S. Chang. Graph transduction via alternating minimization. In *Proceedings of the 25th Annual International Conference on Machine Learning (ICML)*, 2008.

J. Wang and X. Shen. Large margin semi-supervised learning. *Journal of Machine Learning Reserach*, 8:1867–1891, 2007.

Y. Weiss and W. Freeman. On the optimality of solutions of the max-product belief-propagation algorithm in arbitrary graphs. *IEEE Transactions on Information Theory*, 47, 2001.

J. Weston and C. Watkins. Multi-class support vector machines. Technical Report CSD-TR-98-04, Department of Computer Science, Royal Holloway, University of London, 1998.

J. Weston, F. Ratle, and R. Collobert. Deep learning via semi-supervised embedding. In *Proceedings of the 25th Annual International Conference on Machine Learning (ICML)*, 2008.

J. Weston, R. Collobert, F. Sinz, L. Bottou, and V. Vapnik. Inference with the universum. In *Proceedings of the 23rd Annual International Conference on Machine Learning (ICML)*, Pittsburgh, USA, 2006.

M. Wu and B. Schölkopf. Transductive classification via local learning regularization. In *Eleventh International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2007.

E. Xing, A. Ng, M. Jordan, and S. Russell. Distance metric learning with application to clustering with side-information. In *Advances in Neural Information Processing Systems (NIPS)*. MIT Press, Cambridge, MA, 2002.

L. Xu and D. Schuurmans. Unsupervised and semi-supervised multi-class support vector machines. In *AAAI-05, The Twentieth National Conference on Artificial Intelligence*, 2005.

Z. Xu, R. Jin, J. Zhu, I. King, and M. Lyu. Efficient convex relaxation for transductive support vector machine. In J. Platt, D. Koller, Y. Singer, and S. Roweis, editors, *Advances in Neural Information Processing Systems (NIPS)*. MIT Press, Cambridge, MA, 2008.

F. Xue and M. Palmer. The Penn Chinese Treebank: phrase structure annotation of a large corpus. *Natural Language Engineering*, 11(02):207–238, 2005.

L. Yang, R. Jin, and R. Sukthankar. Semi-supervised learning with weakly-related unlabeled data : Towards better text categorization. In D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, editors, *Advances in Neural Information Processing Systems (NIPS)*. MIT Press, Cambridge, MA, 2009.

D. Yarowsky. Unsupervised word sense disambiguation rivaling supervised methods. In *Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics*, pages 189–196, 1995.

K. Yu, S. Yu, and V. Tresp. Blockwise supervised inference on large graphs. In *Proc. of the 22nd ICML Workshop on Learning with Partially Classified Training Data*, Bonn, Germany, August 2005.

S. Yu, K. Yu, V. Tresp, and H.-P. Kriegel. Collaborative ordinal regression. In *Proceedings of the 23rd Annual International Conference on Machine Learning (ICML)*, 2006.

S. Yu, B. Krishnapuram, R. Rosales, H. Steck, and R. B. Rao. Bayesian co-training. In J. Platt, D. Koller, Y. Singer, and S. Roweis, editors, *Advances in Neural Information Processing Systems (NIPS)*. MIT Press, Cambridge, MA, 2008.

J. Zhang, Z. Ghahramani, and Y. Yang. A probabilistic model for online document clustering with application to novelty detection. In *Advances in Neural Information Processing Systems (NIPS)*. MIT Press, Cambridge, MA, 2004.

T. Zhang and R. Ando. Analysis of spectral kernel design based semi-supervised learning. In Y. Weiss, B. Schölkopf, and J. Platt, editors, *Advances in Neural Information Processing Systems (NIPS)*. MIT Press, Cambridge, MA, 2006.

X. Zhang and W. S. Lee. Hyperparameter learning for graph based semi-supervised learning algorithms. In *Advances in Neural Information Processing Systems (NIPS)*. MIT Press, Cambridge, MA, 2006.

D. Zhou and C. Burges. Spectral clustering with multiple views. In *Proceedings of the 24th Annual International Conference on Machine Learning (ICML)*, 2007.

D. Zhou, O. Bousquet, T. Lal, J. Weston, and B. Schölkopf. Learning with local and global consistency. In *Advances in Neural Information Processing Systems (NIPS)*. MIT Press, Cambridge, MA, 2003.

D. Zhou, J. Huang, and B. Schölkopf. Learning from labeled and unlabeled data on a directed graph. In *Proceedings of the 22nd Annual International Conference on Machine Learning (ICML)*, Bonn, Germany, 2005.

D. Zhou, J. Huang, and B. Schölkopf. Learning with hypergraphs: Clustering, classification, and embedding. In *Advances in Neural Information Processing Systems (NIPS)*. MIT Press, Cambridge, MA, 2006.

Y. Zhou and S. Goldman. Democratic co-learing. In *Proceedings of the 16th IEEE International Conference on Tools with Artificial Intelligence (ICTAI 2004)*, 2004.

Z.-H. Zhou, D.-C. Zhan, and Q. Yang. Semi-supervised learning with very few labeled training examples. In *Twenty-Second AAAI Conference on Artificial Intelligence (AAAI-07)*, 2007.

Z.-H. Zhou and M. Li. Semi-supervised regression with co-training. In *International Joint Conference on Artificial Intelligence (IJCAI)*, 2005a.

Z.-H. Zhou and M. Li. Tri-training: exploiting unlabeled data using three classifiers. *IEEE Transactions on Knowledge and Data Engineering*, 17(11):1529–1541, 2005b.

Z.-H. Zhou and J.-M. Xu. On the relation between multi-instance learning and semi-supervised learning. In *Proceedings of the 24th Annual International Conference on Machine Learning (ICML)*, 2007.

J. Zhu, S. Rosset, T. Hastie, and R. Tibshirani. 1-norm support vector machines. In *Neural Information Processing Systems 16*, 2004a.

X. Zhu. Semi-supervised learning literature survey. Technical Report 1530, Department of Computer Sciences, University of Wisconsin, Madison, 2005.

X. Zhu and Z. Ghahramani. Towards semi-supervised classification with Markov random fields. Technical Report CMU-CALD-02-106, Carnegie Mellon University, 2002.

X. Zhu and A. B. Goldberg. Kernel regression with order preferences. In *Twenty-Second AAAI Conference on Artificial Intelligence (AAAI-07)*, 2007.

X. Zhu and A. B. Goldberg. *Introduction to Semi-Supervised Learning*. Morgan & Claypool, 2009.

X. Zhu and A. B. Goldberg. Semi-supervised regression with order preferences. Technical Report 1578, Department of Computer Sciences, University of Wisconsin-Madison, 2006.

X. Zhu and J. Lafferty. Harmonic mixtures: combining mixture models and graph-based methods for inductive and scalable semi-supervised learning. In *Proceedings of the 22nd Annual International Conference on Machine Learning (ICML)*. ACM Press, 2005.

X. Zhu, Z. Ghahramani, and J. Lafferty. Semi-supervised learning using Gaussian fields and harmonic functions. In *Proceedings of the 20th Annual International Conference on Machine Learning (ICML)*, 2003.

X. Zhu, J. Kandola, Z. Ghahramani, and J. Lafferty. Nonparametric transforms of graph kernels for semi-supervised learning. In L. K. Saul, Y. Weiss, and L. Bottou, editors, *Advances in Neural Information Processing Systems (NIPS)*. MIT Press, Cambridge, MA, 2004b.

A. Zien, U. Brefeld, and T. Scheffer. Transductive support vector machines for structured variables. In *Proceedings of the 24th Annual International Conference on Machine Learning (ICML)*, 2007.

M. Zinkevich. Online convex programming and generalized infinitesimal gradient ascent. In *Proceedings of the 20th Annual International Conference on Machine Learning (ICML)*, 2003.