Thesis Proposal
Techniques for Exploiting Unlabeled Data

Mugizi Robert Rwebangira

April 2007

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

*Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy.*

Keywords: semi-supervised, regression, unlabeled data, similarity

Abstract

In many machine learning application domains obtaining labeled data is expensive but obtaining unlabeled data is much cheaper. For this reason there has been growing interest in algorithms that are able to take advantage of unlabeled data. In this proposal we develop several methods for taking advantage of unlabeled data in classification and regression tasks.

Specific contributions include:

- A method for improving the performance of the graph mincut algorithm of Blum and Chawla [10] by taking randomized mincuts. We give theoretical motivation for this approach and we present empirical results showing that randomized mincut tends to outperform the original graph mincut algorithm, especially when the number of labeled examples is very small.

- An algorithm for semi-supervised regression based on manifold regularization using local linear estimators. This is the first extension of local linear regression to the semi-supervised setting. In this proposal we present experimental results on both synthetic and real data and show that this method tends to perform better than methods which only utilize the labeled data.

- An investigation of practical techniques for using the Winnow algorithm (which is not directly kernelizable) together with kernel functions and general similarity functions via unlabeled data. We expect such techniques to be particularly useful when we have a large feature space as well as additional similarity measures that we would like to use together with the original features. This method is also suited to situations where the best performing measure of similarity does not satisfy the properties of a kernel. We present some preliminary experimental results of this approach.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation and Summary

In the modern era two of the most significant trends affecting the use and storage of information are the following:

1. The rapidly increasing speed of electronic microprocessors.
2. The even more rapidly increasing capacity of electronic storage devices.

The latter has allowed the storage of vastly greater amounts of information than was possible in the past and the former has allowed the use of increasingly computationally intensive algorithms to process the collected information.

However, in the past few decades for many tasks the supply of information has outpaced our ability to effectively utilize it. For example in classification we are supplied with pairs of variables (X, Y) and we are asked to come up with a function that predicts the corresponding Y values for a new X. For example we might be given the images of handwritten digits and the corresponding digit that they represent and be asked to learn an algorithm that automatically classifies new images into digits. Such an algorithm has many practical applications, for example the US Postal Service uses a similar algorithm for routing its mail.[33]

The problem is that obtaining the initial training data which we need to learn can be expensive and might even require human intervention to label each example. In the previous example we would need someone to examine every digit in our data set and determine its classification. In this case the work would not require highly skilled labor but it would still be impractical if we have hundreds of thousands of unlabeled examples.

Thus, a natural question is whether and how we can somehow make use of unlabeled examples to aid us in classification. This question has recently been studied with increasing intensity and several algorithms and theoretical insights have emerged.

First, as with all learning approaches we will need to make assumptions about the problem in order to develop algorithms. These assumptions typically involve the relationship between the unlabeled examples and the labels we want to predict. Examples of such assumptions include:

Large Margin Assumption - The data can be mapped into a space such that there exists a linear separator with a“large margin” that separates the (true) positive and (true) negative examples. “Large margin” has a technical definition but intuitively it simply means that the positive and negative an algorithm that uses this assumption.

Graph Partition - The data can be represented as a graph such that the (true) positive and (true) negative examples form which we will discuss in chapter 3 makes this type of assumption.

Cluster Assumptions - Using the given distance metric the (true) positive examples and the (true) negative examples fall into two distinct clusters. This assumption is fairly general and in particular the two previous assumptions can be regarded as special cases of this assumption.

We will discuss these assumptions further in chapter 2 of this proposal.

Furthermore, if we make certain assumptions about the unlabeled data that turn out to not be true in practice not only will the unlabeled data not be helpful, it may actually cause our algorithm to perform worse than it would have if it ignored the unlabeled data. Thus unlabeled data should certainly be incorporated with care in any learning process.

From this discussion it is clear that there probably does not exist any universally applicable semi-supervised learning algorithm which performs better than all other algorithms on problems of interest. The usefulness of any semi-supervised learning algorithm will depend greatly on what kinds of assumptions it makes and whether such assumptions are met in practice. Thus we need to develop a variety of techniques for exploiting unlabeled data and practical experience in addition to theoretical guidance in choosing the best algorithm for specific situations.

In this proposal we develop three such methods for exploiting unlabeled data. First, we present the randomized graph mincut algorithm. This is an extension of the graph mincut algorithm for semi-supervised learning proposed by Blum and Chawla [10]. When the number of labeled examples is small the original graph mincut algorithm has a tendency to give severely unbalanced labelings (i.e. to label almost all examples as positive or to label almost all examples as negative). We show how randomization can be used to address this problem and we present experimental data showing substantially improved performance over the original graph mincut algorithm.

Secondly, we present an algorithm for semi-supervised regression. The use of unlabeled data for regression has not been as well developed as it has been in classification. In semi-supervised regression we are presented with pairs of values (X, Y) where the Y values are real numbers and in addition we have a set of unlabeled examples X'. The task is to estimate the Y value for the unlabeled examples. We present Local Linear Semi-supervised Regression which is a very natural extension of both local Linear Regression and of the Gaussian Fields algorithm for semi-supervised learning of Zhu et. al. [66]. We present some experimental results showing that this algorithm usually performs better than purely supervised methods.

Lastly, we examine the problem of learning with similarity functions. There has been a vast tradition of learning with kernel function in the machine learning community. However kernel functions have strict mathematical definitions (e.g. kernel functions have to be positive semi-definite). Recently there has been growing interest in techniques for learning even when the similarity function does not satisfy the mathematical properties of a kernel function. The connection with semi-supervised learning is that many of these techniques require more examples than the kernel based methods but the additional data can be unlabeled. That is, we can exploit the similarity between examples for learning even when we don't have the labels. Thus if we have large amounts of unlabeled data we would certainly be interested in using suitably defined similarity functions to improve our performance.

In the rest of this chapter we will describe the problems that this proposal addresses in more detail. In chapter 2 we will discuss some of the related work that exists in the literature. In chapters 3,4 and 5 we will describe the work that has been completed on the randomized min-cut algorithm, Local Linear Semi-supervised Regression and learning with similarity functions respectively. In chapter 6 we will discuss the work that we propose to do in the future. Finally in chapter 7, I will describe the expected contributions of the thesis and give a time line for its completion.

1.2 Problem Description

1.2.1 Semi-supervised Classification

The classification problem is central in machine learning and statistics. In this problem we are given as input pairs of variables $(X_1, Y_1), ...(X_m, Y_m)$ where the X_i are objects of the type that we want to classify (for example documents or images) and the Y_i are the corresponding labels of the X_i (for example if the X_i are newspaper articles then the Y_i might indicate whether X_i is an article about machine learning). The goal is to minimize error rate on future examples X whose labels are not known. The special case where Y_i can only have two possible values is known as binary classification. This problem is also called supervised classification to contrast it with semi-supervised classification.

This problem has been extensively studied in the machine learning community and several algorithms have been proposed. A few of the algorithms which gained broader acceptance are perceptron, neural nets, decision trees and support vector machines. We will not discuss supervised classification further in this proposal but there are a number of textbooks which provide a good introduction to these methods. [Tom Mitchell, Duda Hart and Stork, Hastie Tibshirani and Friedman, Luc Devroye, Wasserman I, Wasserman II]

In the semi-supervised classification problem, in addition to labeled examples $(X_1, Y_1), ...(X_m, Y_m)$ we also receive unlabeled examples $X_{m+1}, ...X_n$. Thus we have m unlabeled examples and $n-m$ unlabeled examples.

In this setting we can define two distinct kinds of tasks:

1. To learn a function that takes any X_i and computes a corresponding Y_i.
2. To compute a corresponding Y_i for each of our unlabeled examples without necessarily producing a function that makes a prediction for any X_i. (This is sometimes referred to as *transductive classification*.)

This problem only began to receive extensive attention in the early 90s although several algorithms were known before that. Some of the algorithms that have been proposed for this problem include the Expectation-Maximization algorithm proposed by Dempster, Laird and Rubin[26], the co-training algorithm proposed by Blum and Mitchell[11], the graph mincut algorithm proposed by Blum and Chawla[10], the Gaussian Fields algorithm proposed by Zhu, Gharamani and Lafferty[66] and Laplacian SVM proposed by Sindhwani, Belkin and Niyogi[52].We will discuss these algorithms further in chapter 2 of this proposal.

This area is still the subject of a very active research effort. A number of researchers have attempted to address the question of “Under what circumstances can unlabeled data be useful” from a theoretical point of view [Nina, Castelli, Oles and Zhang, Venkatesh, Lafferty Wasserman unpublished]and there also has been great interest from industrial practitioners who would like to make the best use of their unlabeled data.

1.2.2 Semi-supervised Regression

Regression is a fundamental tool in statistical analysis. At its core regression aims to model the relationship between 2 or more random variable. For example, an economist might want to investigate whether more education leads to an increased income. A natural way to accomplish this is to take number of years of education as the dependent variable and annual income as the independent variable and to use regression analysis to determine their relationship.

Formally, we are given as input $(X_1, Y_1), ...(X_n, Y_n)$ where the X_i are the dependent variables and Y_i are the independent variables. We want to predict for any X the value of the corresponding Y. There are two main types of techniques used to accomplish this:

1. Parametric regression: In this case we assume that the relationship between the variable is of a certain type (e.g. a linear relationship) and we are concerned with learning the parameters for a relationship of that type which best fit the data.

2. Non-parametric regression: In this case we do not make any assumptions about the type of relationship that holds between the variables, but we derive this relationship directly from the data.

Regression analysis is heavily used in the natural sciences and in social sciences such as economics, sociology and political science. A wide variety of regression algorithms are used including linear regression, polynomial regression and logistic regression among the parametric methods and kernel regression and local linear regression among the non-parametric methods. A further discussion of such methods can be found in any introductory statistics textbook. [Wasserman I, Wasserman II, Hogg and Tanis]

In semi-supervised regression in addition to getting the dependent and independent variables X and Y we are also given an addition variable R which indicates whether or not we observe that value of Y. In other words we get data $(X_1, Y_1, R_1), ...(X_n, Y_n, R_n)$ and we observe Y_i only if $R_i = 1$.

We note that the problem of semi-supervised regression is more general than the semi-supervised classification problem. In the latter case the Y_i are constrained to have only a finite number of possible values whereas in regression the Y_i are assumed to be continuous. Hence some algorithms designed for semi-supervised classification (e.g. graph mincut[10]) are not applicable to the more general semi-supervised regression problem. Other algorithms such as Gaussian Fields [66]) are applicable to both problems.

Although semi-supervised regression has received less attention than semi-supervised classification a number of methods have been developed dealing specifically with this problem. These include the transductive regression algorithm proposed by Cortes and Mohri [20]and co-training style algorithms proposed by Zhou and Li [63], Sindhwani et. al.[52] and Brefeld et. al.[14] We will discuss these related approaches in more detail in chapter 2.

1.2.3 Online Learning

In many machine learning algorithms it is often useful to compute a measure of the similarity between two objects. Kernel functions are a popular way of doing this. A kernel function $K(x, y)$ takes two objects and outputs a positive number that is a measure of the similarity of the objects. A kernel function must also satisfy the mathematical condition of positive semi-definiteness.

It follows from Mercer's theorem that any kernel function can also be interpreted as an inner product in some Euclidean vector space. This allows us to take advantage of the representation power of high-dimensional vector spaces without having to explicitly represent our data in such spaces. Due to this insight (known as the "kernel trick") kernel methods have become extremely popular in the machine learning community, resulting in widely used algorithms such as Support Vector Machines. [Nello and Cristianini]

However, sometimes it turns out that a useful measure of similarity does not satisfy the mathematical conditions to be a kernel function. An example is the Smith-Waterman score which is a measure of the alignment of two protein sequences. It is widely used in molecular biology and has been empirically successful in predicting the similarity of proteins. However, the Smith-Waterman score is not a valid kernel function and hence cannot be directly used in algorithms such as Support Vector Machines.

Thus there has been a growing interest in developing more general methods for utilizing similarity functions which may not satisfy the positive semi-definite requirement. Recently, Balcan and Blum [1] proposed a general theory of learning with similarity functions. They give a natural definition of similarity function which contains kernel functions as a sub-class and show that effective learning can be done in this framework.Although the work in this area is still very preliminary, there is strong interest due to the practical benefits of being able to exploit more general kinds of similarity functions.

Chapter 2

Related Work

In the last decade or so there has been a substantial amount of work on exploiting unlabeled data. The survey by Zhu [64] is the most up to date summary of work in this area. The recent book edited by Chapelle et. al. [22] is a good summary of work done up to 2005 and attempts to synthesize the main insights that have been gained. In this chapter we will mainly highlight the work that is most closely related to our own.

2.1 Semi-supervised Classification

As with supervised classification, semi-supervised classification methods fall into two categories

1. Generative methods which attempt to model the statistical distribution that generates the data before making predictions.
2. Discriminative methods which directly make predictions without assumptions about the distribution the data comes from.

2.1.1 Generative Methods

Expectation-Maximization (EM)

Suppose the examples are $(x_1, x_2, x_3, ..., x_n)$ and the labels are $(y_1, y_2, y_3, ...y_l)$. Generative models try to model the class conditional densities $p(x|y)$ and then apply Bayes' rule to compute predictive densities $p(y|x)$.

BAYES' RULE: $p(y|x) = \frac{p(x|y)p(y)}{\int_y p(x|y)p(y)dy}$

In this setting unlabeled data gives us more information about $p(x)$ and we would like to use this information to improve our estimate of $p(y|x)$. If information about $p(x)$ is not useful to estimating $p(y|x)$ (or we do not use it properly) then unlabeled data will not help to improve our predictions.

In particular if our assumptions about the distribution the data comes from are incorrect then unlabeled data can actually degrade our classification accuracy. Cozman and Cohen explore this effect in detail for generative semi-supervised learning models [24].

However, if our generative model for x is correct then any additional information about $p(x)$ will generally be useful. For example suppose that $p(x|y)$ is a gaussian. Then the task becomes to estimate the parameters of the gaussian. This can easily be done with sufficient labeled data, but if there a few labeled examples unlabeled data can greatly improve the estimate. Castelli and Cover [19] provide a theoretical analysis of this scenario.

A popular algorithm to use in the above scenario is the Expectation-Maximization algorithm proposed by Dempster, Laird and Rubin [26]. EM is an iterative algorithm that can be used to compute Maximum-Likelihood estimates of parameters when some of the relevant information is missing. It is guaranteed to converge and is fairly fast in practice. However it is only guaranteed to converge to a local minima.

An advantage of generative approaches is that knowledge about the structure of a problem can be naturally incorporated by modelling it in the generative model. The work by Nigam et. al. [46] on modelling text data is an example of this approach. But as stated before, if the assumptions are incorrect unlabeled data can lead to worse inferences than completely ignoring the unlabeled data.

2.1.2 Discriminative Methods

As stated before, in semi-supervised classification we wish to exploit the relationship between $p(x)$ and $p(y|x)$. with generative methods we assume that the distribution generating the data has a certain form and the unlabeled data helps us to learn the parameters of the distribution more accurately. Discriminative methods do not bother to try and model the distribution generating the data $p(x)$, hence in order to use unlabeled data we have to make “a priori” assumptions on the relationship between $p(x)$ and $p(y|x)$.

We can categorize discriminative methods for semi-supervised learning by the kind of assumption they make on relationship between the distribution of examples and the conditional distribution of the labels. In this survey we will focus on a family of algorithms that use what is known as the Cluster Assumption.

The Cluster Assumption

The cluster assumption posits a simple relationship between $p(x)$ and $p(y|x)$: $p(y|x)$ should change slowly in regions where $p(x)$ has high density. Informally this is equivalent to saying that examples that are “close” to each other should have “similar” labels.

Hence, gaining information about $p(x)$ gives information about the high density region (clusters) in which examples should be given the same label. This can also be viewed as a “reordering” of the hypothesis space: We give preference to those hypotheses which do not violate the cluster assumption.

The cluster assumption can be realized in variety of ways and leads to a number of different algorithms. Most prominent are **graph based methods** in which a graph is constructed that contains all the labeled and unlabeled examples as nodes and the weight of an edge between nodes indicates the similarity of the corresponding examples.

We note that graph based methods are inherently transductive although it is easy to extend them to the inductive case by taking the predictions of the semi-supervised classification as training data and then using a non-parametric supervised classification algorithm such as k-nearest neighbor on the new example. In this case classification time will be significantly faster than training time. Another option is to rerun the entire algorithm when presented with a new example.

Graph Based Methods

Graph Mincut

The graph mincut algorithm proposed by Blum and Chawla [10] was one of the earliest graph based methods to appear in the literature. The essential idea of this algorithm is to convert semi-supervised classification into an *s*-*t* mincut problem. The algorithm is as follows:

1. Construct a graph joining examples which have similar labels. The edges may be weighted or unweighted.

2. Connect the positively labeled examples to a “source” node with “high-weight” edges. Connect the negatively labeled examples to a “sink” node with “high-weight” edges.

3. Obtain an *s*-*t* minimum cut of the graph and classify all the nodes connected to the “source” as positive examples and all the nodes connected to the sink as negative examples.

There is considerable freedom in the construction of the graph. Properties that are empirically found to be desirable are that the graph should be connected, but that it shouldn’t be too dense.

As stated the algorithm only applies to binary classification, but one can easily imagine extending it to general classification by taking a multi-way cut. However, this would entail a significant increase in the running time as multi-way cut is known to be NP-complete. We note that there has been substantial work in the image-segmentation literature on using multi-way cuts (e.g. Boykov et. al. [13].)

This algorithm is attractive because it is simple to understand and easy to implement. However, a significant problem is that it can often return very "unbalanced" cuts. More precisely, if the number of labeled examples is small, the *s-t* minimum cut may chop off a very small piece of the graph and return this is as the solution. Furthermore, there is no known natural way to "demand" balanced cut without running into a significantly harder computational problem. (The Spectral Graph Transduction algorithm proposed by Joachims is one attempt to do this efficiently).

In chapter 3 of this proposal we report on techniques for overcoming this problem of unbalanced cuts and show significant improvement in results compared to the original graph mincut algorithm.

Gaussian Fields Method [66]

This central idea of this algorithm is to reduce semi-supervised classification to finding the minimum energy of a certain Gaussian Random Field.

The algorithm is as follows:

1. Compute a weight W_{ij} between all pairs of examples.
2. Find real values f that minimize the energy functional $E(f) = \frac{1}{2}\sum_{i,j} w_{ij}(f(i) - f(j))^2$ (where the value of the labeled f_i's are fixed.)
3. Assign each (real) f_i to one of the discrete labels.

It turns out that the solution to step (2) is closely connected to random walks, electrical networks and spectral graph theory. For example, if we think of each edge as a resistor, it is equivalent to placing a battery with positive terminal connected to the positive labeled examples, negative terminal connected to the negative labeled examples and measuring voltages at the unlabeled points.

Further, this method can also be viewed as a continuous relaxation of the graph mincut method. More importantly the solution can be computed using only matrix operations for the cost of inverting a $u \times u$ matrix (where u is the number of labeled examples).

A major advantage of this method is that is fairly straightforward to implement. In addition, unlike graph-mincut it can be generalized to the multi-label case without a significant increase in complexity.

However, note that as stated it could still suffer from an "unbalanced" labelings. Zhu et. al. [66] report on using a "Class Mass Normalization" heuristic to force the unlabeled examples to have the same class proportions as the labeled examples.

Laplacian SVM[4]

This main idea in this method is to take the SVM objective function and add an extra regularization penalty that penalizes similar examples that have different labels.

More precisely the solution for SVM can be stated as

$f^* = argminf \in \mathcal{H}_\mathcal{K} = \frac{1}{l}\sum_{i=1}^{l} V(x_i, y_i, f) + \gamma||f||_K^2$

$V(x_i, y_i, f)$ is some loss function such as squared loss $(y_i - f(x_i))^2$ or soft margin loss $max\{0, 1 - y_i f(x_i)\}$.

The second term is a regularization that imposes smoothness condition on possible solutions.

In Laplacian SVM the solution is the following:

$f^* = argminf \in \mathcal{H}_\mathcal{K} = \frac{1}{l}\sum_{i=1}^{l} V(x_i, y_i, f) + \gamma_A||f||_K^2 + \frac{\gamma_I}{(u+l)^2}\sum_{i,j} w_{ij}(f(i) - f(j))^2$

The extra regularization term imposes an additional smoothness condition over both the labeled and unlabeled data. We note that the extra regularization term is the same as the objective function in the Gaussian Fields method.

An advantage of this method is that it easily extends to the inductive case since the representer theorem still applies.

$f^*(x) = \sum_{i=1}^{l+u} \alpha_i K(x_i, x)$. (Representer Theorem)

On the other hand, implementation of this method will necessarily be more complicated since it involves solving a quadratic program.

Transductive SVM (TSVM)

Transductive SVM was first proposed by Vapnik [[58],[59]] as a natural extension of the SVM algorithm to unlabeled data. The idea is very simple: instead of simply finding a hyperplane that maximizes the margin on the labeled examples we want to find a hyperplane that maximizes the margin on the labeled and unlabeled examples.

Unfortunately this simple extension fundamentally changes the nature of the optimization problem. In particular the original SVM leads to a convex optimization problem which has a unique solution and can be solved efficiently. TSVM does not lead to a convex optimization problem (due to the non-linear constraints imposed by the unlabeled data) and in particular there is no known method to solve it efficiently.

However there are encouraging empirical results: Thorsten Joachims proposed an algorithm that works by first obtaining a supervised SVM solution then doing a coordinate descent to improve the objective function. It showed significant improvement over purely supervised methods and scaled up to 100,000 examples. [37]

Die Bie and Cristianini proposed a semi-definite programming relaxation of the TSVM[[7][8]]. They then further proposed an approximation of this relaxation and showed that at least in some cases they attain better performance than the version of TSVM proposed by Joachims. However, their method is not able to scale to more than 1000 examples.

TSVM is very attractive from a theoretical perspective as it inherits most of the justifications of SVM and the large-margin approach, however as of now it is still very challenging from an implementation point of view.

Spectral Graph Transducer (SGT)[36]

The central idea in this method is to ensure that the proportions of positive and negative examples is the same in the labeled and unlabeled sets. As we noted this is an issue in several methods based on the Cluster Assumption such as graph mincut and Gaussian Fields.

SGT addresses this issue by reducing the problem to an unconstrained ratio-cut in a graph with additional constraints to take into account the labeled data. Since the resulting problem is still NP-Hard, the real relaxation of this problem is used for the actual solution. It turns out that this can be solved efficiently.

Joachims reports some impressive experimental results in comparison with TSVM, kNN and SVM. Furthermore, the algorithm has an easy to understand and attractive intuition. However, the reality is that it is not (so far) possible to solve the original problem optimally and it is not clear what the quality of the solution of the relaxation is. Some experimental results by Blum et. al. [12] indicate that SGT is very sensitive to fine tuning of its parameters.

2.2 Semi-supervised Regression

While some semi-supervised classification such as Gaussian Fields can be directly applied to semi-supervised regression without any modifications, there has been relatively little work directly targeting semi-supervised regression. It is not clear if this is due to lack of interest from researchers or lack of demand from practitioners.

Correspondingly the theoretical insights and intuitions are not as well developed for semi-supervised regression as they are for semi-supervised classification. In particular so far no taxonomy for semi-supervised regression methods has been proposed that corresponds to the taxonomy for semi-supervised classification methods.

Still, the overall task remains the same: to use information about the distribution of x to improve our estimates of $p(y|x)$. To this end, we need to make an appropriately useful assumption. The Manifold assumption is one such candidate: We assume that the data lies along a low dimensional manifold.

An example of this would be if the true function we are trying to estimate is a 3 dimensional spiral. If we only have a few labeled examples, it would be hard to determine the structure of the entire manifold. However, if we have sufficiently large amount of unlabeled data, the manifold becomes much easier to determine. The Manifold assumption also implies the Smoothness assumption: Examples which are close to each other, have similar labels. In other words we expect the function to not “jump” suddenly.

2.2.1 Transductive Regression algorithm of Cortes and Mohri[20]

The transductive regression algorithm of Cortes and Mohri minimizes an objective function of the following form:

$\mathcal{E}(h) = ||w||^2 + C\sum_{i=1}^{m}(h(x_i) - y_i)^2 + C'\sum_{i=m+1}^{m+u}(h(x_i) - \hat{y}_i)^2$

Here the hypothesis h is a linear function $w \in \mathcal{F}, x \in \mathcal{X}, h(x) = w \cdot \Phi(x)$

Where $\mathcal{F}$ is a vector space endowed with a norm,the examples are drawn from $\mathcal{X}$, Φ is a feature from $\mathcal{X}$ to $\mathcal{F}$ and C and C' are regularization parameters.

Hence the task is to estimate the vector w of weights.

A strong attraction of this method is that it is quite computationally efficient: It essentially only requires the inversion of an $D \times D$ matrix where D is the dimension of $\mathcal{F}$ the space into which the examples are mapped. Cortes and Mohri [20] report impressive empirical results on various regression data sets.

However, it is not clear how much performance is sacrificed by assuming the hypothesis is of the form $h(x) = w \cdot \Phi(x)$. It would be interesting to compare with a purely non-parametric method.

2.2.2 COREG (Zhou and Li) [63]

The key idea of this method is to apply co-training to the semi-supervised regression task. The algorithm uses two k-nearest neighbor regressors with different distance metrics, each of which labels the unlabeled data for the other regressor. The labeling confidence is estimated through consulting the influence of the labeling of unlabeled examples on the labeled ones.

Zhou and Li report positive experimental results on mostly synthetic data sets. However, since this was the first paper to deal with semi-supervised regression, they were not able to compare with more recent techniques.

The concept of applying co-training to regression is attractive, however it is not clear what unlabeled data assumptions are being exploited.

2.2.3 Co-regularization (Sindhwani et. al.)[52]

This technique also uses co-training but in a regularization framework.

More precisely the aim is to learn two different classifiers by optimizing an objective function that simultaneously minimizes their training error and their disagreement with each other.

Brefeld et. al. [14] use the same idea, they are also propose a fast approximation that scales linearly in the number of training examples.

2.3 Online Learning

In spite of (or maybe because of) the wide popularity of kernel based learning methods in the past decade, there has been relatively little work on learning with similarity functions that are not kernels.

The paper by Balcan and Blum [1] was the first to rigorously analyze this framework. The main contribution was to define a notion of good similarity function for learning which was intuitive and included the usual notion of a large margin kernel function.

Recently Srebro [54] gave an improved analysis with tighter bounds on the relation between large margin kernel functions and good similarity functions in the Balcan-Blum sense. In particular he showed that large margin kernel functions remain good when used with a hinge loss. He also gives examples where using a kernel function as a similarity function can produce worse margins although his results do not imply that the margin will always be worse if a kernel is used as a similarity function.

A major practical motivation for this line of research is the existence of domain specific similarity functions (typically defined by domain experts) which are known to be successful but which do not satisfy the definition of a kernel function. The Smith-Waterman score used in biology is a typical example of such a similarity function. The Smith-Waterman score is a measure of local alignment between protein sequences. It is considered by biologist to be the best measure of homology (similarity) between two protein sequences.

However, it does not satisfy the definition of a kernel function and hence cannot be used in kernel based methods. To deal with this issue Vert et. al. [60] construct a convolution kernel to "mimic" the behavior of the Smith-Waterman score. Vert et. al. report promising experimental results, however it is highly likely that the original Smith-Waterman score provides a better measure of similarity than the kernelized version. Hence techniques that use the original similarity measure might potentially have superior performance.

In this work we will focus on using similarity functions with online algorithms like Winnow. One motivation for this is in our approach similarity functions are most useful if we have large amounts of unlabeled data. For large data sets we need fast algorithms such as Winnow.

Another advantage of Winnow specifically is that it can learn well even in presence of irrelevant features. This is important in learning with similarities as we will typically create several new features for each examples and only a few of the generated features may be relevant to the classification task.

Winnow was proposed by Littlestone [43] who analyzed some of its basic properties. Blum [9] and Dagan [25] demonstrated that Winnow could be used in real world tasks such as calendar scheduling and text classification. More recent experimental work has shown that versions of Winnow can be competitive with the best offline classifiers such as SVM and Logistic Regression [Bekkerman [3], Cohen & Carvalho [17] [18].]

Chapter 3

Completed Work: Randomized Mincuts

In this chapter we will describe the randomized mincut algorithm for semi-supervised classification. We will give some background and motivation for this approach and also give some experimental results.

3.1 Introduction

If one believes that "similar examples ought to have similar labels," then a natural approach to using unlabeled data is to combine nearest-neighbor prediction—predict a given test example based on its nearest labeled example—with some sort of self-consistency criteria, e.g., that similar *unlabeled* examples should, in general, be given the same classification. The graph mincut approach of [10] is a natural way of realizing this intuition in a transductive learning algorithm. Specifically, the idea of this algorithm is to build a graph on all the data (labeled and unlabeled) with edges between examples that are sufficiently similar, and then to partition the graph into a positive set and a negative set in a way that (a) agrees with the labeled data, and (b) cuts as few edges as possible. (An edge is "cut" if its endpoints are on different sides of the partition.)

The graph mincut approach has a number of attractive properties. It can be found in polynomial time using network flow; it can be viewed as giving the most probable configuration of labels in the associated Markov Random Field (see Section 3.2); and, it can also be motivated from sample-complexity considerations, as we discuss further in Section 3.4.

However, it also suffers from several drawbacks. First, from a practical perspective, a graph may have many minimum cuts and the mincut algorithm produces just one, typically the "leftmost" one using standard network flow algorithms. For instance, a line of n vertices between two labeled points s and t has $n-1$ cuts of size 1, and the leftmost cut will be especially unbalanced. Second, from an MRF perspective, the mincut approach produces the most probable joint labeling (the MAP hypothesis), but we really would rather label nodes based on their *per-node* probabilities (the Bayes-optimal prediction). Finally, from a sample-complexity perspective, if we could average over many small cuts, we could improve our confidence via PAC-Bayes style arguments.

Randomized mincut provides a simple method for addressing a number of these drawbacks. Specifically, we repeatedly add artificial random noise to the edge weights,[1] solve for the minimum cut in the resulting graphs, and finally output a fractional label for each example corresponding to the fraction of the time it was on one side or the other in this experiment. This is not the same as sampling directly from the MRF distribution, and is also not the same as picking truly random minimum cuts in the original graph, but those problems appear to be much more difficult computationally on general graphs (see Section 3.2).

A nice property of the randomized mincut approach is that it easily leads to a measure of confidence on the predictions; this is lacking in the deterministic mincut algorithm, which produces a single partition of the data. The confidences allow us to compute accuracy-coverage curves, and we see that on many data sets the randomized mincut algorithm exhibits good accuracy-coverage performance.

We also discuss design criteria for constructing graphs likely to be amenable to our algorithm. Note that some graphs simply do not have small cuts that match any low-error solution; in such graphs, the mincut approach will likely fail even with randomization. However, constructing the graph in a way that is very conservative in producing edges can alleviate many of these problems. For instance, we find that a very simple minimum spanning tree graph does quite well across a range of data sets.

PAC-Bayes sample complexity analysis [44] suggests that when the graph has many small cuts consistent with the labeling, randomization should improve generalization performance. This analysis is supported in experiments with data sets such as handwritten digit recognition, where the algorithm results in a highly accurate classifier. In cases where the graph does not have small cuts for a given classification problem, the theory also suggests, and our experiments confirm, that randomization may not help. We present experiments on several different data sets that indicate both the strengths and weaknesses of randomized mincuts, and also how this approach compares with the semi-supervised learning schemes of [66] and [36]. For the case of MST-graphs, in which the Markov random field probabilities *can* be efficiently calculated exactly, we compare to that method as well.

In the following sections we will give some background on Markov random fields, describe our algorithm more precisely as well as our design criteria for graph construction, provide sample-complexity analysis motivating some of our design decisions, and finally give some experimental results.

[1]We add noise only to existing edges and do not introduce new edges in this procedure.

3.2 Background and Motivation

Markov random field models originated in statistical physics, and have been extensively used in image processing. In the context of machine learning, what we can do is create a graph with a node for each example, and with edges between examples that are similar to each other. A natural energy function to consider is

$$E(f) = \frac{1}{2}\sum_{i,j} w_{ij}|f(i) - f(j)| = \frac{1}{4}\sum_{i,j} w_{ij}(f(i) - f(j))^2$$

where $f(i) \in \{-1, +1\}$ are binary labels and w_{ij} is the weight on edge (i, j), which is a measure of the similarity between the examples. To assign a probability distribution to labelings of the graph, we form a random field

$$p_\beta(f) = \frac{1}{Z}\exp\left(-\beta E(f)\right)$$

where the partition function Z normalizes over all labelings. Solving for the lowest energy configuration in this Markov random field will produce a partition of the entire (labeled and unlabeled) data set that maximally optimizes self-consistency, subject to the constraint that the configuration must agree with the labeled data.

As noticed over a decade ago in the vision literature [30], this is equivalent to solving for a minimum cut in the graph, which can be done via a number of standard algorithms. [10] introduced this approach to machine learning, carried out experiments on several data sets, and explored generative models that support this notion of self-consistency.

The minimum cut corresponds, in essence, to the MAP hypothesis in this MRF model. To produce Bayes-optimal predictions, however, we would like instead to sample directly from the MRF distribution.

Unfortunately, that problem appears to be much more difficult computationally on general graphs. Specifically, while random labelings can be efficiently sampled *before* any labels are observed, using the well-known Jerrum-Sinclair procedure for the Ising model [34], after we observe the labels on some examples, there is no known efficient algorithm for sampling from the conditional probability distribution; see [27] for a discussion of related combinatorial problems.

This leads to two approaches:

1. Try to approximate this procedure by adding random noise into the graph
2. Make sure the graph is a tree, for which the MRF probabilities can be calculated exactly using dynamic programming.

Here,we will consider both.

3.3 Randomized Mincuts

The randomized mincut procedure we consider is the following. Given a graph G constructed from the data set, we produce a collection of cuts by repeatedly adding random noise to the edge weights and then solving for the minimum cut in the perturbed graph.

In addition, now that we have a collection of cuts, we remove those that are highly unbalanced. This step is justified using a simple ϵ-cover argument (see Section 3.4), and in our experiments, any cut with less than 5% of the vertices on one side is considered unbalanced.[2] Finally, we predict based on a majority vote over the remaining cuts in our sample, outputting a confidence based on the margin of the vote. We call this algorithm "Randomized mincut with sanity check" since we use randomization to produce a distribution over cuts, and then throw out the ones that are obviously far from the true target function.

In many cases this randomization can overcome some of the limitations of the plain mincut algorithm. Consider a graph which simply consists of a line, with a positively labeled node at one end and a negatively labeled node at the other end with the rest being unlabeled. Plain mincut may choose from any of a number of cuts, and in fact the cut produced by running network flow will be either the leftmost or rightmost one depending on how it is implemented. Our algorithm will take a vote among all the mincuts and thus we will end up using the middle of the line as a decision boundary, with confidence that increases linearly out to the endpoints.

It is interesting to consider for which graphs our algorithm produces a true uniform distribution over minimum cuts and for which it does not. To think about this, it is helpful to imagine we collapse all labeled positive examples into a single node s and we collapse all labeled negative examples into a single node t. We can now make a few simple observations. First, a class of graphs for which our algorithm *does* produce a true uniform distribution are those for which all the s-t minimum cuts are disjoint, such as the case of the line above. Furthermore, if the graph can be decomposed into several such graphs running in parallel between s and t ("generalized theta graphs" [16]), then we get a true uniform distribution as well. That is because any minimum s-t cut must look like a tuple of minimum cuts, one from each graph, and the randomized mincut algorithm will end up choosing at random from each one.

On the other hand, if the graph has the property that some minimum cuts overlap with many others and some do not, then the distribution may not be uniform. For example, Figure 3.1 shows a case in which the randomized procedure gives a much higher weight to one of the cuts than it should ($\geq 1/6$ rather than $1/n$).

Looking at Figure 3.1 ,in the top graph, each of the n cuts of size 2 has probability $1/n$ of being minimum when random noise is added to the edge lengths. However, in the bottom graph this is not the case. In particular, there is a constant probability that the noise added to

[2]With only a small set of labeled data, one cannot in general be confident that the true class probabilities are close to the observed fractions in the training data, but one *can* be confident that they are not extremely biased one way or the other.

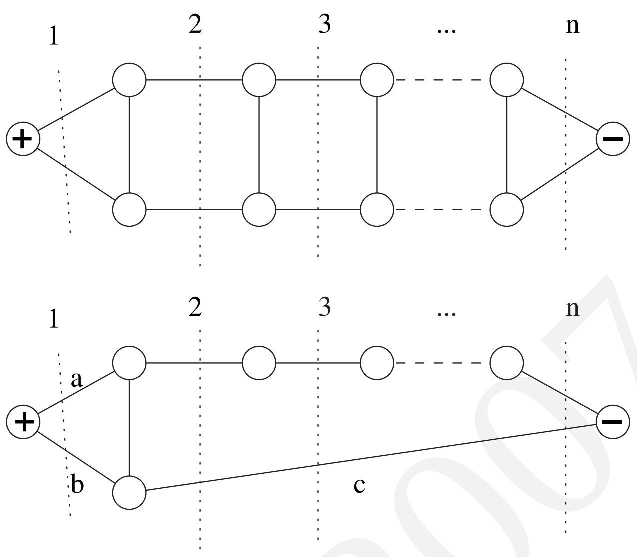

Figure 3.1: A case where randomization will not uniformly pick a cut

edge c exceeds that added to a and b combined (if a, b, c are picked at random from $[0, 1]$ then $\Pr(c > a+b) = 1/6$). This results in the algorithm producing cut $\{a, b\}$ no matter what is added to the other edges. Thus, $\{a, b\}$ has a much higher than $1/n$ probability of being produced.

3.4 Sample complexity analysis

3.4.1 The basic mincut approach

From a sample-complexity perspective, we have a transductive learning problem, or (roughly) equivalently, a problem of learning from a known distribution. Let us model the learning scenario as one in which first the graph G is constructed from data without any labels (as is done in our experiments) and then a few examples at random are labeled. Our goal is to perform well on the rest of the points. This means we can view our setting as a standard PAC-learning problem over the uniform distribution on the vertices of the graph. We can now think of the mincut algorithm as motivated by standard Occam bounds: if we describe a hypothesis by listing the edges cut using $O(\log n)$ bits each, then a cut of size k can be described in $O(k \log n)$ bits.[3] This means we need only $O(k \log n)$ labeled examples to be confident in a consistent cut of k edges (ignoring dependence on ϵ and δ).

In fact, we can push this bound further: [39], studying the problem of detecting failures in networks, shows that the VC-dimension of the class of cuts of size k is $O(k)$. Thus, only $O(k)$ labeled examples are needed to be confident in a consistent cut of k edges. [41] reduce this further to $O(k/\lambda)$ where λ is the size of the *global* minimum cut in the graph (the minimum number

[3]This also assumes the graph is connected — otherwise, a hypothesis is not uniquely described by the edges cut.

of edges that must be removed in order to separate the graph into two nonempty pieces, without the requirement that the labeled data be partitioned correctly).

One implication of this analysis is that if we imagine data is being labeled for us one at a time, we can plot the size of the minimum cut found (which can only increase as we see more labeled data) and compare it to the global minimum cut in the graph. If this ratio grows slowly with the number of labeled examples, then we can be confident in the mincut predictions.

3.4.2 Randomized mincut with "sanity check"

As pointed out by [36], minimum cuts can at times be very unbalanced. From a sample-complexity perspective we can interpret this as a situation in which the cut produced is simply not small enough for the above bounds to apply given the number of labeled examples available. From this point of view, we can think of our mincut extension as being motivated by two lines of research on ways of achieving rules of higher confidence.

The first of these are PAC-Bayes bounds [42, 44]. The idea here is that even if no single consistent hypothesis is small enough to inspire confidence, if many of them are "pretty small" (that is, they together have a large prior if we convert our description language into a probability distribution) then we can get a better confidence bound by randomizing over them.

Even though our algorithm does not necessarily produce a true uniform distribution over all consistent minimum cuts, our goal is simply to produce as wide a distribution as we can to take as much advantage of this as possible. Results of [28] show furthermore that if we weight the rules appropriately, then we can expect a lower error rate on examples for which their vote is highly biased.

Again, while our procedure is at best only an approximation to their weighting scheme, this motivates our use of the bias of the vote in producing accuracy/coverage curves. We should also note that recently Hanneke[31] has carried out a much more detailed version of the analysis that we have only hinted at here.

The second line of research motivating aspects of our algorithm is work on bounds based on ϵ-cover size, e.g., [5]. The idea here is that suppose we have a known distribution D and we identify some hypothesis h that has many similar hypotheses in our class with respect to D. Then if h has a high error rate over a labeled sample, it is likely that *all* of these similar hypotheses have a high true error rate, *even if some happen to be consistent with the labeled sample*.

In our case, two specific hypotheses we can easily identify of this form are the "all positive" and "all negative" rules. If our labeled sample is even reasonably close to balanced — e.g., 3 positive examples out of 10 — then we can confidently conclude that these two hypotheses have a high error rate, and throw out *all highly unbalanced cuts*, even if they happen to be consistent with the labeled data.

For instance, the cut that simply separates the three positive examples from the rest of the graph is consistent with the data, but can be ruled out by this method.

This analysis then motivates the second part of our algorithm in which we discard all highly unbalanced cuts found before taking majority vote. The important issue here is that we can confidently do this even if we have only a very small labeled sample. Of course, it is possible that by doing so, our algorithm is never able to find a cut it is willing to use. In that case our algorithm halts with failure, concluding that the data set is not one that is a good fit to the biases of our algorithm. In that case, perhaps a different approach such as the methods of [36] or [66] or a different graph construction procedure is needed.

3.5 Graph design criteria

For a given distance metric, there are a number of ways of constructing a graph. In this section, we briefly discuss design principles for producing graphs amenable to the graph mincut algorithm. These then motivate the graph construction methods we use in our experiments.

First of all, the graph produced should either be connected or at least have the property that a small number of connected components cover nearly all the examples. If t components are needed to cover a $1 - \epsilon$ fraction of the points, then clearly any graph-based method will need t labeled examples to do well.[4]

Secondly, for a mincut-based approach we would like a graph that at least has some small balanced cuts. While these may or may not correspond to cuts consistent with the labeled data, we at least do not want to be dead in the water at the start. This suggests conservative methods that only produce edges between very similar examples.

Based on these criteria, we chose the following two graph construction methods for our experiments.

MST: Here we simply construct a minimum spanning tree on the entire data set. This graph is connected, sparse, and furthermore has the appealing property that it has no free parameters to adjust. In addition, because the exact MRF per-node probabilities *can* be exactly calculated on a tree, it allows us to compare our randomized mincut method with the exact MRF calculation.

δ**-MST:** For this method, we connect two points with an edge if they are within a radius δ of each other. We then view the components produced as supernodes and connect them via

[4]This is perhaps an obvious criterion but it is important to keep in mind. For instance, if examples are uniform random points in the 1-dimensional interval $[0, 1]$, and we connect each point to its nearest k neighbors, then it is not hard to see that if k is fixed and the number of points goes to infinity, the number of components will go to infinity as well. That is because a local configuration, such as two adjacent tight clumps of k points each, can cause such a graph to disconnect.

an MST. [10] used δ such that the largest component had half the vertices (but did not do the second MST stage). To produce a more sparse graph, we choose δ so that the largest component has $1/4$ of the vertices.

Another natural method to consider would be a k-NN graph, say connected up via a minimum spanning tree as in δ-MST. However, experimentally, we find that on many of our data sets this produces graphs where the mincut algorithm is simply not able to find even moderately balanced cuts (so it ends up rejecting them all in its internal "sanity-check" procedure). Thus, even with a small labeled data set, the mincut-based procedure would tell us to choose an alternative graph-creation method.

3.6 Experimental Analysis

We compare the randomized mincut algorithm on a number of data sets with the following approaches:

PLAIN MINCUT: Mincut without randomization.

GAUSSIAN FIELDS: The algorithm of [66].

SGT: The spectral algorithm of [36].

EXACT: The exact Bayes-optimal prediction in the MRF model, which can be computed efficiently in trees (so we only run it on the MST graphs).

Below we present results on handwritten digits, portions of the 20 newsgroups text collection, and various UCI data sets.

3.6.1 Handwritten Digits

We evaluated randomized mincut on a data set of handwritten digits originally from the Cedar Buffalo binary digits database [33]. Each digit is represented by a 16 X 16 grid with pixel values ranging from 0 to 255. Hence, each image is represented by a 256-dimensional vector.

For each size of the labeled set, we perform 10 trials, randomly sampling the labeled points from the entire data set. If any class is not represented in the labeled set, we redo the sample.

One vs. Two: We consider the problem of classifying digits, "1" vs. "2" with 1128 images. Results are reported in Figure 3.2. We find that randomization substantially helps the mincut procedure when the number of labeled examples is small, and that randomized mincut and the Gaussian field method perform very similarly. The SGT method does not perform very well on this data set for these graph-construction procedures. (This is perhaps an unfair comparison, because our graph-construction procedures are based on the needs of the mincut algorithm, which may be different than the design criteria one would use for graphs for SGT.)

Odd vs. Even: Here we classify 4000 digits into Odd vs. Even. Results are given in Figure 3.3. On the MST graph, we find that Randomized mincut, Gaussian fields, and the exact MRF calculation all perform well (and nearly identically). Again, randomization substantially

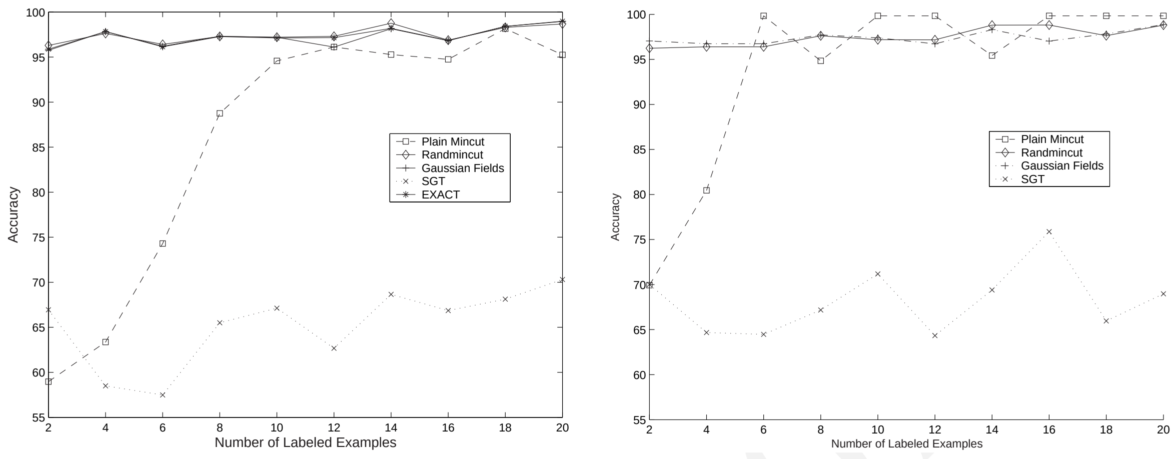

Figure 3.2: “1” vs “2” on the digits data set with the MST graph (left) and $\delta_{\frac{1}{4}}$ graph (right).

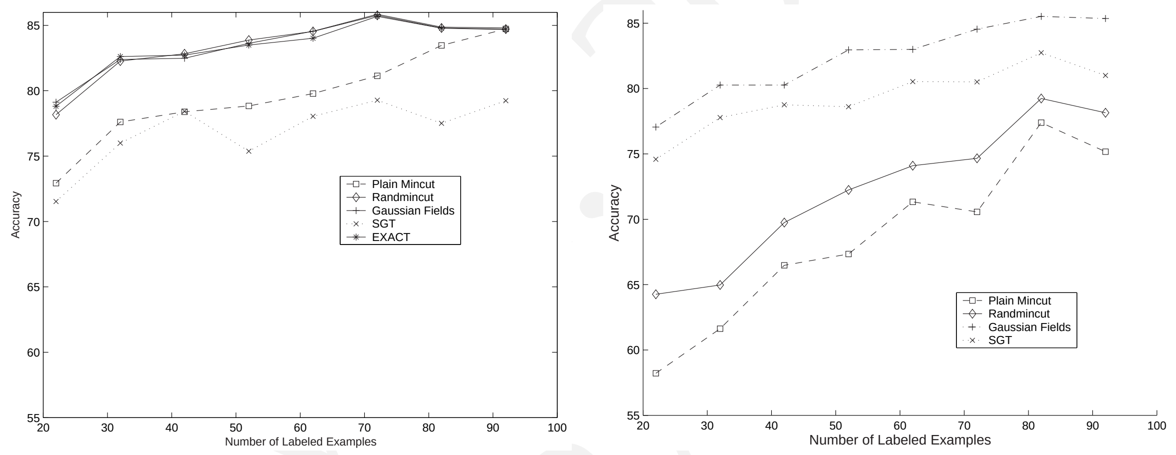

Figure 3.3: Odd vs. Even on the digits data set with the MST graph (left) and $\delta_{\frac{1}{4}}$ graph (right).

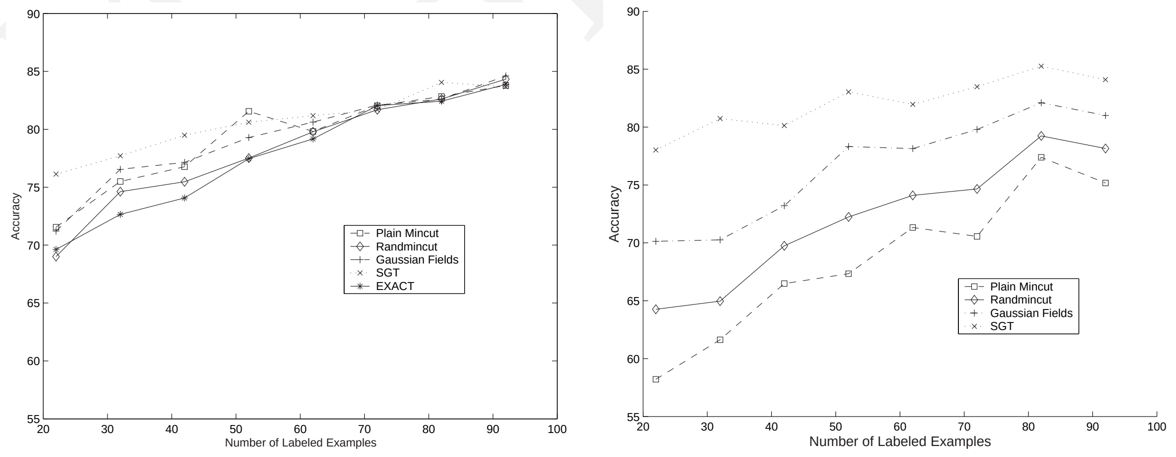

Figure 3.4: PC vs MAC on the 20 newsgroup data set with MST graph (left) and $\delta_{\frac{1}{4}}$ graph (right).

helps the mincut procedure when the number of labeled examples is small. On the δ-MST graph, however, the mincut-based procedures perform substantially worse, and here Gaussian fields and SGT are the top performers.

In both data sets, the randomized mincut algorithm tracks the exact MRF Bayes-optimal predictions extremely closely. Perhaps what is most surprising, however, is how good performance is on the simple MST graph. On the Odd vs. Even problem, for instance, [66] report for their graphs an accuracy of 73% at 22 labeled examples, 77% at 32 labeled examples, and do not exceed 80% until 62 labeled examples.

3.6.2 20 newsgroups

We performed experiments on classifying text data from the 20-newsgroup data sets, specifically PC versus MAC (see Figure 3.4). Here we find that on the MST graph, all the methods perform similarly, with SGT edging out the others on the smaller labeled set sizes. On the δ-MST graph, SGT performs best across the range of labeled set sizes. On this data set, randomization has much less of an effect on the mincut algorithm.

DATA SET	$\lvert L\rvert\&\lvert U\rvert$	FEAT.	GRAPH	MINCUT	RAND MINCUT	GAUSSIAN	SGT	EXACT
			MST	92.0	90.9	90.6	90.0	90.6
VOTING	45+390	16	$\delta_{\frac{1}{4}}$	92.3	91.2	91.0	85.9	—
			MST	86.1	89.4	92.2	89.0	92.4
MUSH	20+1000	22	$\delta_{\frac{1}{4}}$	94.3	94.2	94.2	91.6	—
			MST	78.3	77.8	79.2	78.1	83.9
IONO	50+300	34	$\delta_{\frac{1}{4}}$	78.8	80.0	82.8	79.7	—
			MST	63.5	64.0	63.7	61.8	63.9
BUPA	45+300	6	$\delta_{\frac{1}{4}}$	62.9	62.9	63.5	61.6	—
			MST	65.7	67.9	66.7	67.7	67.7
PIMA	50+718	8	$\delta_{\frac{1}{4}}$	67.9	68.8	67.5	68.2	—

Table 3.1: Classification accuracies of basic mincut, randomized mincut, Gaussian fields, SGT, and the exact MRF calculation on data sets from the UCI repository using the MST and $\delta_{\frac{1}{4}}$ graph.

3.6.3 UCI Data sets

We conducted experiments on various UC Irvine data sets; see Table 3.1. Here we find all the algorithm perform comparably.

3.6.4 Accuracy Coverage Tradeoff

As mentioned earlier, one motivation for adding randomness to the mincut procedure is that we can use it to set a confidence level based on the number of cuts that agree on the classification of a particular example. To see how confidence affects prediction accuracy, we sorted the examples by

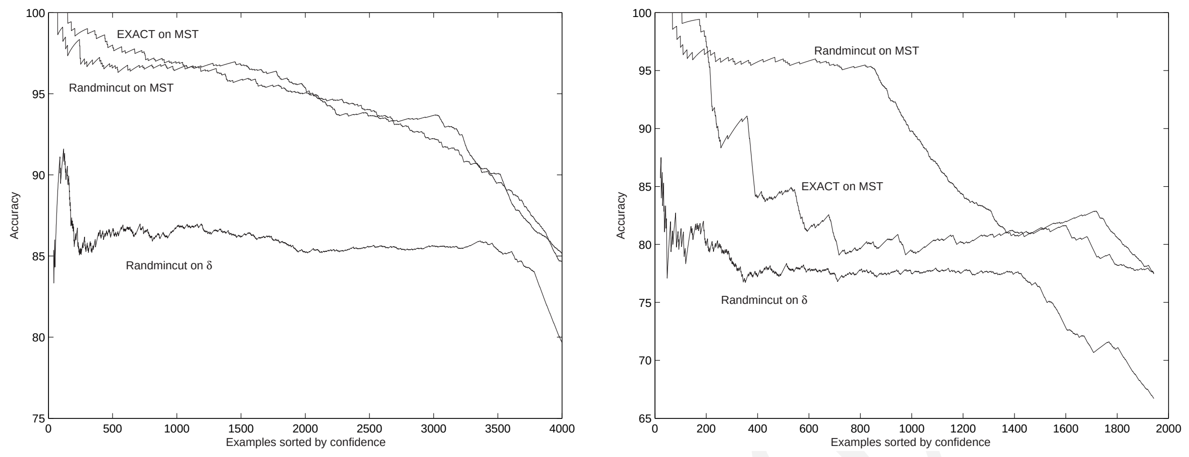

Figure 3.5: Accuracy coverage tradeoffs for randomized mincut and EXACT. Odd vs. Even (left) and PC vs. MAC (right).

confidence and plotted the cumulative accuracy. Figure 3.5 shows accuracy-coverage tradeoffs for Odd-vs-Even and PC-vs-MAC. We see an especially smooth tradeoff for the digits data, and we observe on both data sets that the algorithm obtains a substantially lower error rate on examples on which it has high confidence.

3.6.5 Examining the graphs

To get a feel for why the performance of the algorithms is so good on the MST graph for the digits data set, we examined the following question. Suppose for some $i \in \{0, \ldots, 9\}$ you remove all vertices that are not digit i. What is the size of the largest component in the graph remaining? This gives a sense of how well one could possibly hope to do on the MST graph if one had only one labeled example of each digit. The result is shown in Figure 3.6. Interestingly, we see that most digits have a substantial fraction of their examples in a single component. This partly explains the good performance of the various algorithms on the MST graph.

3.7 Conclusion

The randomized mincut algorithm addresses several shortcomings of the basic mincut approach, improving performance especially when the number of labeled examples is small, as well as providing a confidence score for accuracy-coverage curves. We can theoretical motivate this approach from a sample complexity and Markov Random Field perspective.

The experimental results support the applicability of the randomized mincut algorithm to various settings. In the experiments done so far, our method allows mincut to approach, though it tends not to beat, the Gaussian field method of [66]. However, mincuts have the nice property that we can apply sample-complexity analysis, and furthermore the algorithm can often easily tell when it is or is not appropriate for a data set based on how large and how unbalanced the cuts

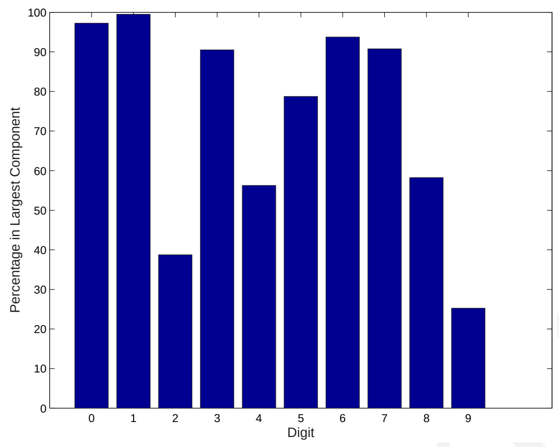

Figure 3.6: MST graph for Odd vs. Even: percentage of digit i that is in the largest component if all other digits were deleted from the graph.

happen to be.

The exact MRF per-node likelihoods can be computed efficiently on trees. It would be interesting if this can be extended to larger classes of graphs.

Chapter 4

Completed Work: Local Linear Semi-supervised Regression

4.1 Semi-supervised Regression

In this chapter we will describe the Local Linear Semi-supervised Regression algorithm. We will give some motivation and background for this approach, formulate and solve it as an optimization problem and present some experimental results.

4.2 Motivation

In formulating semi-supervised classification algorithms an often useful motivating idea is the Cluster Assumption: the assumption that the data will naturally cluster into clumps that have the same label. This notion of clustering does not readily apply to regression, but we can make a somewhat similar "smoothness" assumption: we expect the value of the regression function to not "jump" or change suddenly. In both cases we expect *examples that are close to each other to have similar values.*

A very natural way to instantiate this assumption in semi-supervised regression is by finding estimates $\hat{m}(x)$ that minimize the following objective function:

$$\sum_{i,j} w_{ij}(\hat{m}(x_i) - \hat{m}(x_j))^2$$

where $\hat{m}(x_i)$ is the *estimated* value of the function at example x_i and w_{ij} is a measure of the similarity between examples x_i and x_j.

This is exactly the objective function minimized by the Gaussian Fields algorithm proposed by Zhu, Gharamani and Lafferty [66].

This algorithm has several attractive properties:

1. The solution can be computed in closed form by simple matrix operations.
2. It has interesting connections to Markov Random Fields, electrical networks,spectral graph theory and random walks.

However, it suffers from at least one major drawback when used in regression: It is "locally constant." This means that it will tend to assign the same value to all the example near a particular labeled example and hence produce "flat neighborhoods."

While this behavior is desirable in classification, it is often undesirable in regression application where we frequently assume that the true function is "locally linear." By locally linear we mean that(on some sufficiently small scale) the value of an example is a "linear interpolation" of the value of its closest neighbors. (From a mathematical point of view local linearity is a consequence of a function being differentiable.)

Hence if our function is of this type then a "locally constant" estimator will not provide good estimates and we would prefer to use an algorithm that incorporates a local linearity assumption.

The supervised analogue of Gaussian Fields is Weighted Kernel Regression (also known as the Nadaraya-Watson estimator) which minimizes the following objective function:

$$\sum_i w_i (y_i - \hat{m}(x))^2$$

where $\hat{m}(x)$ is the value of the function at example x,y_i is the value of x_i and w_i is a measure of the similarity between x and x_i.

In the supervised case, there already exists an estimator that has the desired property: Local Linear Regression, which finds β_x so as to minimize the following objective function:

$$\sum_{i=1}^{n} w_i (y_i - \beta_x^T X_{xi})^2$$

with

$$X_{xi} = \begin{pmatrix} 1 \\ x_i - x \end{pmatrix}$$

This is the same as kernel regression if we set $\hat{m}(x) = \beta_x^T X_{xi}$ and we see that the "local linearity" property is "baked into" the objective function.

Hence, a suitable goal is to derive a local linear version of the Gaussian Fields algorithm. Equivalently, we want a semi-supervised version of the Local Linear estimator.

In the remainder of this chapter we will give some background on non-parametric regression, describe the Local Linear Semi-supervised Regression algorithm and show the results of some experiments.

4.3 Background

The general problem of estimating a function from data has been extensively studied in the statistics community. There are two broad classes of methods that are used: (i) Parametric (ii) Non-parametric.

4.3.1 Parametric Regression methods

These approaches assumes that the function that is being estimated is of a particular type and then try to estimate the parameters of the function so that it will best fit the observed data.

For example, we may assume that the function we seek is linear but the observations have been corrupted with Gaussian noise:

$$y = \beta^T x + \epsilon_i \text{ (with } \epsilon_i \sim N(0, \sigma^2))$$

Parametric methods have some advantages compared to non-parametric methods:

1. They are usually easier to analyze mathematically.
2. They usually require less data in order to learn a good model.
3. They are typically less computationally intensive.

4.3.2 Non-Parametric Regression methods

These approaches do not assume that the function we are trying to estimate is of a specific type. i.e given

$$y_i = m(x_i) + \epsilon_i$$

the goal is to estimate the value of $m(x)$ at each point.

The main advantage of non-parametric approaches is that they are more flexible than non-parametric methods and hence they are able to accurately represent broader classes of functions.

4.3.3 Linear Smoothers

Linear smoothers are a class of non-parametric methods in which the function estimates are a linear function of the response variable:

$$\hat{y} = Ly$$

where $\hat{y}$ are the new estimates, y are the observations and L is a matrix which may be constructed based on the data.

Linear smoothers include most commonly used non-parametric regression algorithms and in particular all the algorithms we have discussed so far are linear smoothers:

Weighted Kernel Regression

The objective is to find $\hat{m}(x)$ that minimizes the least squares error

$$\sum_i w_i (y_i - \hat{m}(x))^2$$

The minimizer of this objective function is

$$\hat{m}(x) = \frac{\sum_i w_i y_i}{\sum_i w_i} = Ly$$

Local Linear Regression

The objective is to find β_x that minimizes the least squares error

$$\sum_{i=1}^{n} w_i (y_i - \beta_x^T X_{xi})^2$$

where

$$X_{xi} = \begin{pmatrix} 1 \\ x_i - x \end{pmatrix}$$

The minimizer of the objective function is

$$\hat{\beta}_x = (X_x^T W_x X_x)^{-1} X_x^T W_x y$$

where X_x is the $n \times (d+1)$ matrix (X_{xi}^T) and the matrix W_x is the $n \times n$ diagonal matrix $diag(w_i)$.

The local linear estimate of $m(x)$ is

$$\hat{m}(x) = e_1^T (X_x^T W_x X_x)^{-1} X_x^T W_x y = Ly$$

Gaussian Fields

The objective is to find f that minimizes the energy functional

$$\mathcal{E}(f) = \sum_{i,j} w_{ij}(f_i - f_j)^2$$

It can be shown that

$$\mathcal{E}(f) = f^T \Delta f$$

where $\Delta = D - W$ is the *combinatorial graph Laplacian* of the data.

If f_L denotes the observed labels and f_U denotes the unknown labels then the minimizer of the energy functional is

$$f_U = \Delta_{UU}^{-1} \Delta_{UL} f_L = Ly$$

where Δ_{UU} and Δ_{UL} are the relevant submatrices of the graph Laplacian.

4.4 Local Linear Semi-supervised Regression

Our goal is to derive a semi-supervised analogue of Local Linear Regression so we want it have the following properties.

1. It should fit a linear function at each point like Local Linear Regression.
2. The estimate for a particular point should depend on the estimates for all the other example like Gaussian Fields.

Let X_i and X_j be two examples and β_i and β_j be the local linear fits at X_i and X_j respectively.

Let

$$X_{ji} = \begin{pmatrix} 1 \\ X_i - X_j \end{pmatrix}$$

Then $X_{ji}^T \beta_j$ is the estimated value at X_i using the local linear fit at X_j. Thus the quantity

$$(\beta_{i0} - X_{ji}^T \beta_j)^2$$

is the squared difference between the smoothed estimate at X_i and the estimated value at X_i using the local fit at X_j.

We can take the sum of this quantity of all pairs of examples as the quantity we want to minimize:

$$\Omega(\beta) = \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} w_{ij}(\beta_{i0} - X_{ji}^T \beta_j)^2$$

Lemma 1.1 The manifold regularization functional $\Omega(\beta)$ can be written as the quadratic form

$$\Omega(\beta) = \beta^T \Delta \beta$$

where the local linear Laplacian $\Delta = [\Delta_{ij}]$ is the $n \times n$ block matrix with $(d+1) \times (d+1)$ blocks $\Delta_{ij} = diag(D_i) - [W_ij]$ where

$$D_i = \frac{1}{2} \sum_j w_{ij}(e_1 e_1^T + X_{ij} X_{ij}^T)$$

and

$$W_{ij} = w_{ij} \begin{pmatrix} 1 & (X_j^T - X_i^T) \\ (X_j - X_i) & 0 \end{pmatrix}$$

Proof:
Let n be the number of examples.
Let d be the number of dimensions.
Let X be a $d \times n$ matrix (the data).
Let W be a ***symmetric*** $n \times n$ matrix. (the similarity between X_i and X_j)
Let β be a $n \times (d+1)$ length vector. (the coefficients we want to learn).
Let β_i be the β_{id+1} to $\beta_{i(d+1)}$ coefficients in β. (the coefficients for X_i)
Let X_i be $[X_{i1} \ \dots \ X_{id}]^T$ (The i^{th} column of X)
Let X_{ij} be $[1 \ (X_i - X_j)^T]^T$
Let e_1 be $[1 \ 0 \ 0 \dots \ 0]^T$
Let d_i be $\sum_j W_{ij}$

Starting with the objective function:

$$\Omega(\beta) = \sum_i \sum_j W_{ij}(X_{ii}^T B_i - X_{ij}^T B_j)^2 \text{ (by definition)}$$

$$= \sum_i \sum_j W_{ij} B_i^T X_{ii} X_{ii}^T B_i - \sum_i \sum_j 2W_{ij} B_i^T X_{ii} X_{ij}^T B_j + \sum_i \sum_j W_{ij} B_j^T X_{ij} X_{ij}^T B_j \text{ (expanding)}$$

$$\sum_i \sum_j W_{ij} B_i^T X_{ii} X_{ii}^T B_i = \sum_i B_i^T d_i X_{ii} X_{ii}^T B_i = \mathcal{B}^T \Delta_1 \mathcal{B}$$

Where $[(\Delta_1)_{ii}] = d_i X_{ii} X_{ii}^T$

Let

$$S = \sum_i \sum_j 2W_{ij} B_i^T X_{ii} X_{ij}^T B_j = \sum_i \sum_j 2B_i^T W_{ij} X_{ii} X_{ij}^T B_j$$

$$= \sum_i \sum_j 2B_j^T W_{ji} X_{jj} X_{ji}^T B_i = \sum_i \sum_j 2B_i^T W_{ij} X_{ji} X_{jj}^T B_j$$

$$S = \frac{1}{2}(S+S) = \frac{1}{2} \sum_i \sum_j 2B_i^T W_{ij} (X_{ii} X_{ij}^T + X_{ji} X_{jj}^T) B_j = \mathcal{B}^T \Delta_2 \mathcal{B}$$

Where $[(\Delta_2)_{ij}] = W_{ij}(X_{ii}X_{ij}^T + X_{ji}X_{jj}^T)$

$$\sum_i \sum_j W_{ij} B_j^T X_{ij} X_{ij}^T B_j = \sum_j B_j^T (\sum_i W_{ij} X_{ij} X_{ij}^T) B_j = \mathcal{B}^T \Delta_3 \mathcal{B}$$

Where $[(\Delta_3)_{ii}] = \sum_j W_{ij} X_{ji} X_{ji}^T$

$$\therefore \Omega(\beta) = \sum_i \sum_j W_{ij} (X_{ii}^T B_i - X_{ij}^T B_j)^2 = \mathcal{B}^T \Delta \mathcal{B}$$

where $\Delta = diag(D_i) - [\mathcal{W}_{ij}]$ and

$$diag(D_i) = \Delta_1 + \Delta_3$$

$$[\mathcal{W}_{ij}] = \Delta_2$$

■

Lemma 1.2.

Let $\mathcal{R}_\gamma(\beta)$ be the manifold regularized risk functional

$$\mathcal{R}_\gamma(\beta) = \frac{1}{2} \sum_{j=1}^{n} \sum_{R_i=1} w_{ij} (Y_i - X_{ji}^T \beta_j)^2 + \frac{\gamma}{2} \Omega(\beta)$$

$$= \frac{1}{2} \sum_{j=1}^{n} (Y_i - X_j \beta_j)^T W_j (Y_i - X_j \beta_j) + \frac{\gamma}{2} \beta^T \Delta \beta$$

The minimizer of this risk can be written in closed form as

$$\hat{\beta}(\gamma) = (diag(X_j^T W_j X_j) + \gamma \Delta)^{-1})(X_1^T W_1 Y, \ldots, X_1^T W_1 Y)^T$$

Proof.

$$\frac{1}{2}\sum_{j=1}^{n}\sum_{R_i=1}^{n} W_{ij}(Y_i - X_{ji}^T B_j)^2 + \frac{\gamma}{2}\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} W_{ij}(B_{i0} - X_{ji}^T B_j)^2$$

$$= \frac{1}{2}\sum_{j=1}^{n}(Y - X_j B_j)^T W_j (Y - X_j B_j) + \frac{\gamma}{2}\mathcal{B}^T \Delta \mathcal{B}$$

$$= \frac{1}{2}\sum_{j=1}^{n} Y^T W_j Y - \frac{1}{2}\sum_{j=1}^{n} B_j^T X_j^T W_j Y - \frac{1}{2}\sum_{j=1}^{n} Y^T W_j X_j B_j + \frac{1}{2}\sum_{j=1}^{n} B_j^T X_j^T W_j X_j B_j + \frac{\gamma}{2}\mathcal{B}^T \Delta \mathcal{B}$$

$$= \frac{1}{2} Y^T (\sum_{j=1}^{n} W_j) Y - B^T [X_j^T W_j Y] + \frac{1}{2}\mathcal{B}^T diag(X_j^T W_j X_j)\mathcal{B} + \frac{\gamma}{2}\mathcal{B}^T \Delta \mathcal{B}$$

Let $Q = diag(X_j^T W_j X_j), P = [X_j^T W_j Y], C = \frac{1}{2} Y^T (\sum_{j=1}^{n} W_j) Y$

$$= C - \mathcal{B}^T P + \frac{1}{2}\mathcal{B}^T Q \mathcal{B} + \frac{\gamma}{2}\mathcal{B}^T \Delta \mathcal{B}$$

Differentiating with respect to $\mathcal{B}$ and setting to zero:

$$= -P + Q\mathcal{B} + \gamma \Delta \mathcal{B} = 0$$

$$=> \mathcal{B} = (Q + \gamma \Delta)^{-1} P$$

$$\therefore \hat{\beta}(\gamma) = (diag(X_j^T W_j X_j) + \gamma \Delta)^{-1})(X_1^T W_1 Y, \ldots, X_1^T W_1 Y)^T$$

■

4.5 Experimental Results

To understand the behavior of the algorithm we performed some experiments on both synthetic and real data.

4.5.1 Algorithms

We compared two purely supervised algorithms and with Local Semi-supervised Regression.

WKR - Weighted Kernel Regression

LLR - Local Linear Regression

LLSR - Locally Linear Semisupervised Regression

4.5.2 Parameters

There are two free parameters that we have to set for LLSR (and one (kernel bandwidth) for WKR and LLR).

h - kernel bandwidth

γ - Amount of Semisupervised Smoothing

4.5.3 Error metrics

LOOCV MSE - Leave-One-Out-Cross-Validation Mean Squared Error. This is what we actually try to optimize (analogous to training error).

MSE - This is the true Mean Squared Error between our predictions and the true function (analogous to test error).

4.5.4 Computing LOOCV

Since we do not have access to the MSE we pick the parameters so as to minimize the LOOCV MSE. Computing the LOOCV MSE in a naive way would be very computationally expensive since we would have to run the algorithm $O(n)$ times.

Fortunately for a linear smoother we can compute the LOOCV MSE by running the algorithm only once. More precisely if $\hat{y} = Ly$ then

$$\text{LOOCV MSE} = \sum_{i=1}^{n} \left(\frac{y_i - \hat{y}_i}{1 - L_{ii}}\right)$$

4.5.5 Automatically selecting parameters

We experimented with a number of different ways of picking the parameters. In these experiments we used a form of coordinate descent.

Picking one parameter

To pick one parameter we just reduce the bandwidth until there is no more improvement in LOOCV MSE.

1. Initially set bandwidth to 1 and compute LOOCV MSE.
2. Set $h = h/2$ and compute the resulting LOOCV MSE.
3. If the LOOCV MSE decreases then go back to step 2 else go to step 4
4. Output the h which had the lowest LOOCV MSE.

Picking two parameters

To pick two parameters we succesively halve the parameter which yields the biggest decrease in LOOCV MSE.

1. Initially set both bandwidth and smoothing parameter to 1 and compute LOOCV MSE.
2. Set $h = h/2$ while leaving γ alone and compute LOOCV MSE.
3. Set $\gamma = \gamma/2$ while leaving h alone and compute LOOCV MSE.
4. If either steps 2 or 3 decreased the LOOCV MSE then choose the setting which had the lower LOOCV MSE and go back to step 2 else go to step 5.
5. Output the parameter setting which had the lowest MSE.

These procedures are a crude form of gradient descent.
Although they are not guaranteed to be optimal they are (somewhat) effective in practise.

4.5.6 Synthetic Data Set:Doppler

The Doppler function is a popular function for testing regression algorithms.
Interval: 0.5 to 1.5
Data Points: Sampled uniformly in the interval. (Default $= 800$)
labeled Data Points: Sampled uniformly from the data points. (Default $= 80$).
RED = Estimated values

BLACK - labeled examples

True function: $y = \frac{1}{x} \sin \frac{15}{x}$

$\sigma^2 = 0.1$ (Noise)

Weighted kernel Regression

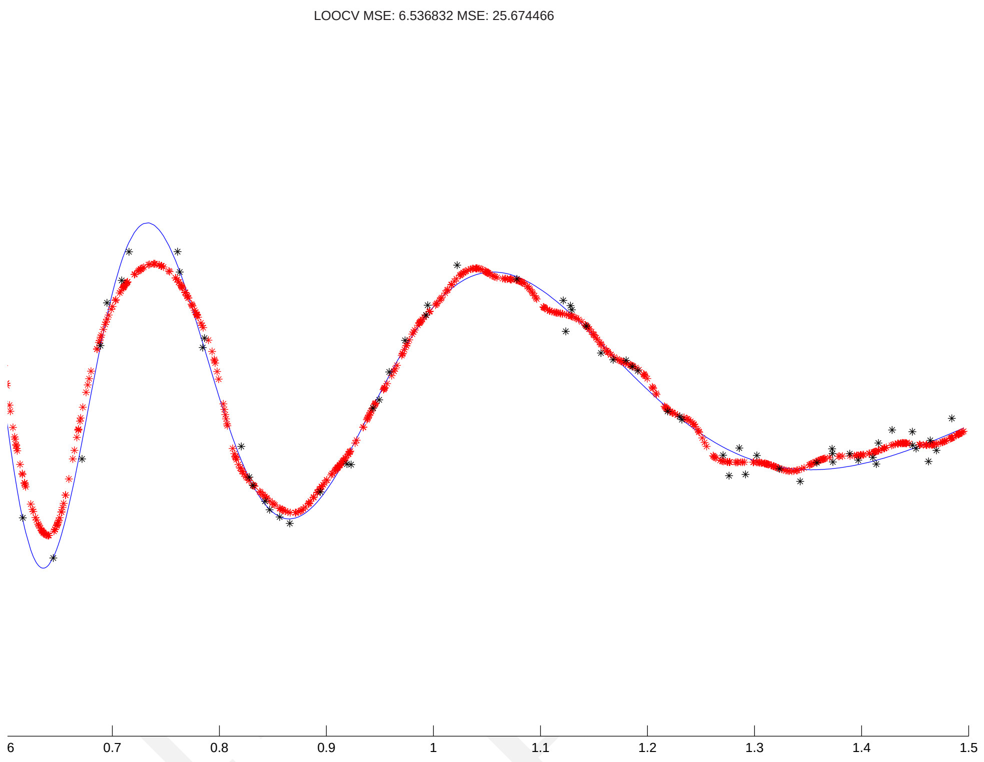

Figure 4.1: WKR on the Doppler example, $h = \frac{1}{128}$

Discussion

There is significant bias on the left boundary and at the peaks and valleys. In this case it seems like a local linear assumption might be more appropriate.

Local Linear Regression

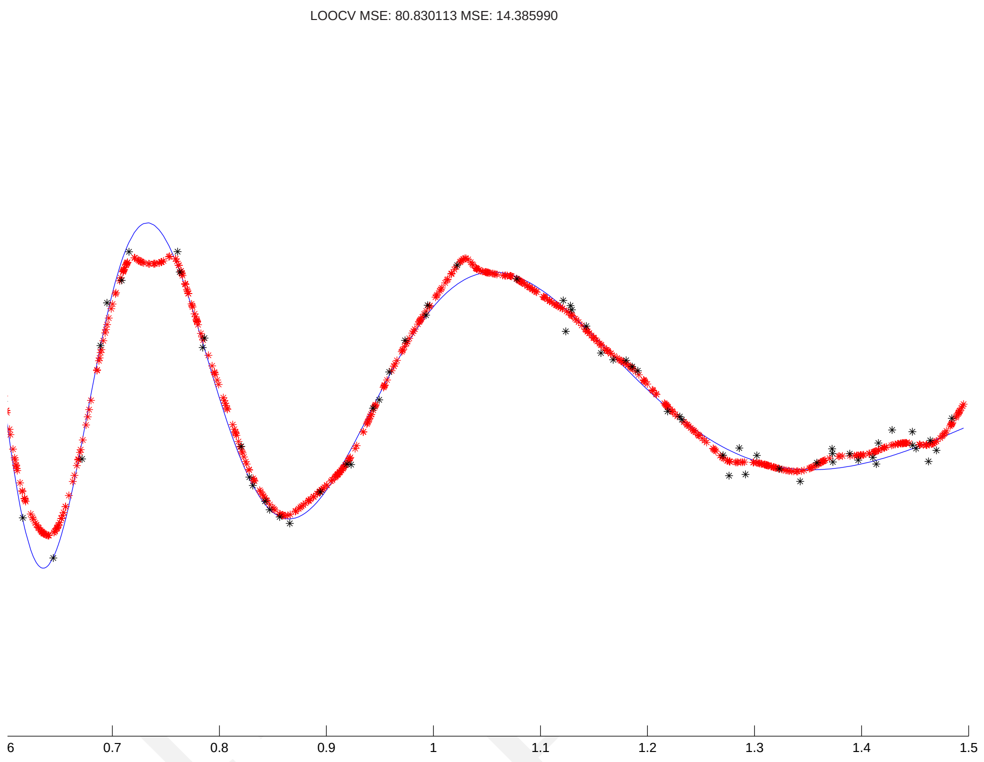

Figure 4.2: LLR on the Doppler example, $h = \frac{1}{128}$

Discussion

There is less bias on the left boundary but there seems to be over fitting on the rightmost peak. It seems like more smoothing is needed.

Local Linear Semi-supervised Regression

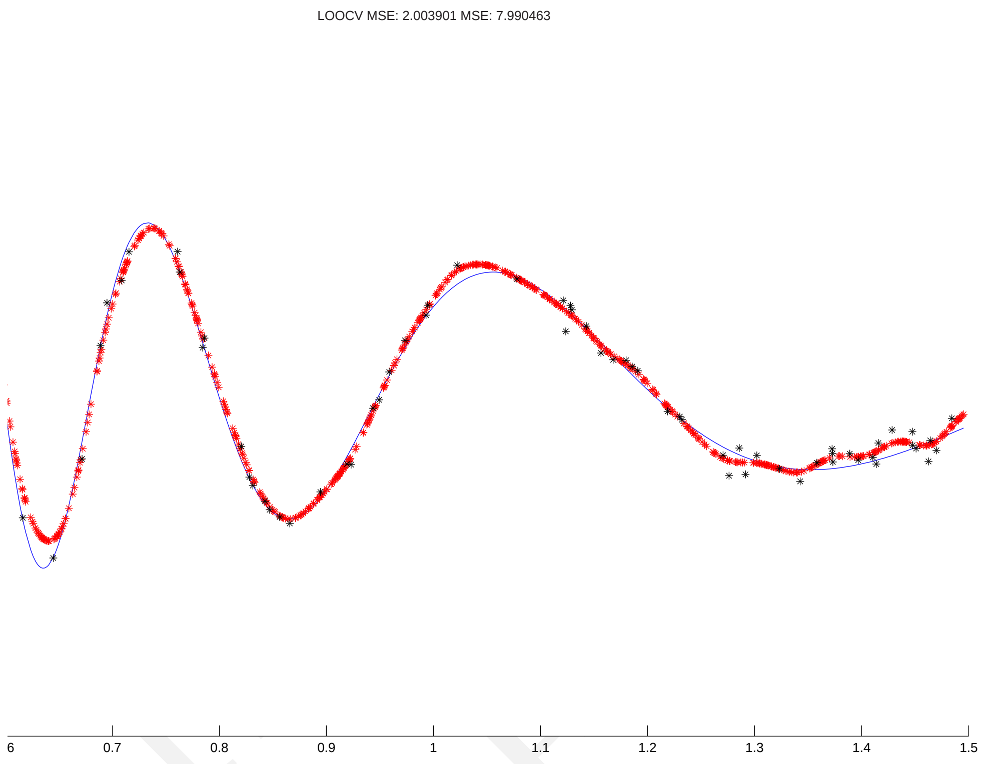

Figure 4.3: LLSR on the Doppler example, $h = \frac{1}{512}, \gamma = 1$

Discussion

Although the fit is not perfect, it is the best of the 3. It manages to avoid boundary bias and fits most of the peaks and valleys.

4.5.7 Real Data Set: CO_2

The following data set shows the Carbon dioxide concentration in the atmosphere over the last two centuries and is obtained from the WorldWatch Institute

Number of points:58

labeled points: 20

labeled Data Points: We uniformly select a subset of the examples and label them and treat the rest as unlabeled data.

RED = Estimated values

BLACK - labeled examples

Weighted Kernel Regression

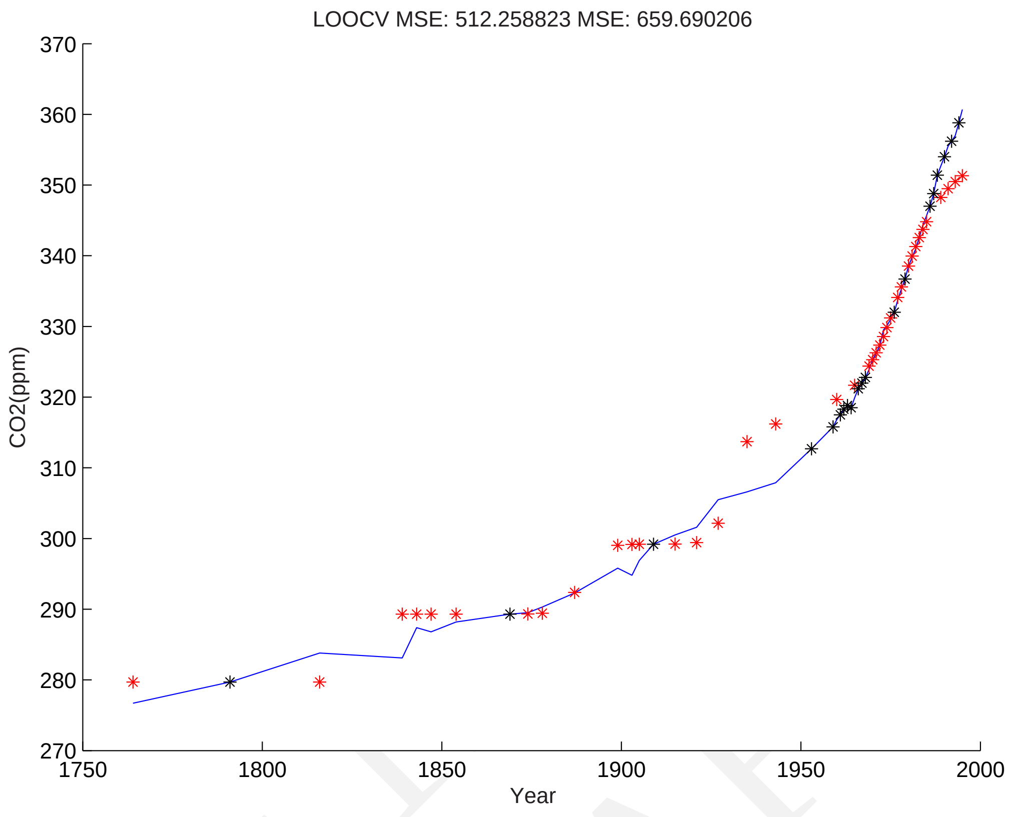

Figure 4.4: WKR on the CO_2 data set, $h = 10$

Discussion

First we can note that there are a lot more examples in recent years than in the distant past. The effect of the local constant assumption is visible in the parallel bands that are visible on the left. Also there is strong boundary bias on the right. Given the form of the function it looks as though a local linear assumption might perform better on this data set.

Local Linear Regression

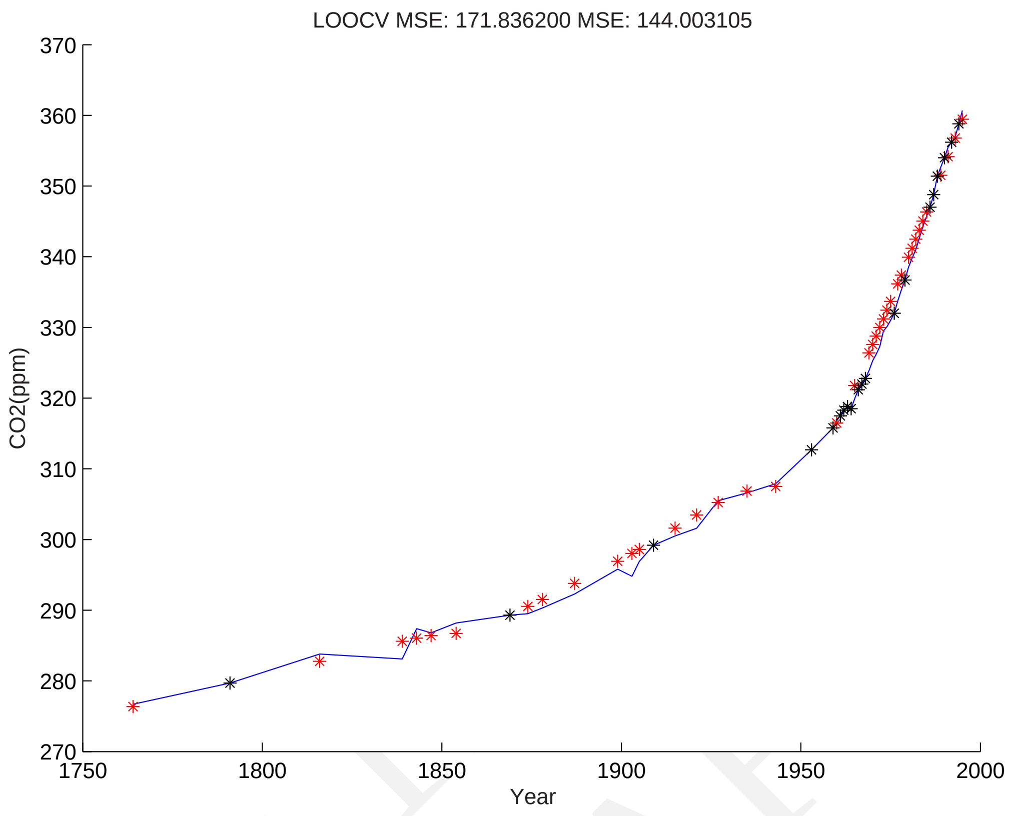

Figure 4.5: LLR on the CO_2 data set, $h = 20$

Discussion

The fit is significantly better and avoids most of the bias that occurred with WKR.

Local Linear Semi-supervised Regression

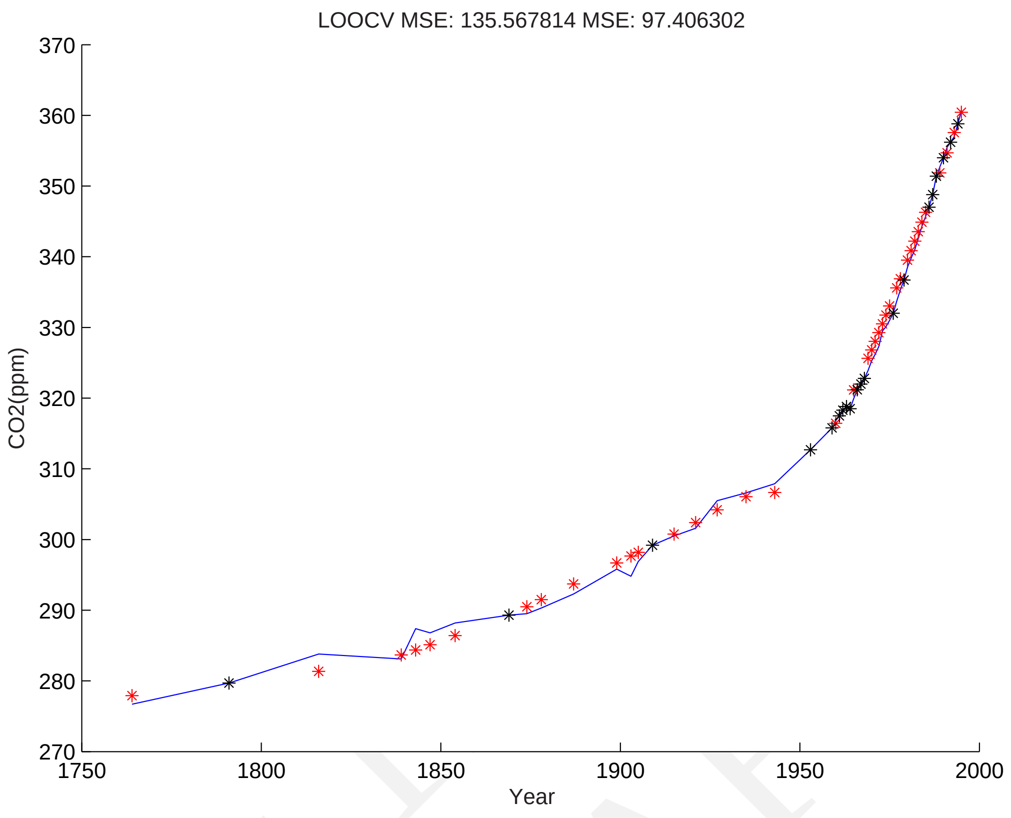

Figure 4.6: LLSR on the CO_2 data set, $h = 10, \gamma = \frac{1}{32}$

Discussion

The overall quality of the fit is similar to LLR although it looks somewhat smoother and the overall error is smaller.

4.5.8 Real Data Set: Auto-mpg

This data set shows how many miles a car will travel per gallon of fuel.
Number of points: 392
labeled points: 40
Dimension of examples: 7

Algorithm	MSE	LOOCV MSE	h	γ
Weighted Kernel Regression	772.128635	7191.596193	200	-
Local Linear Regression	857.647773	4457.216611	400	-
Local Linear Semi-supervised Regression	700.937	4051.308668	400	1

Table 4.1: Performance on the auto-mpg data set

4.5.9 Real Data: Housing

This data set shows the price of a house based on various factors.
Number of points: 392
labeled points: 40
Dimension of examples: 13

Algorithm	MSE	LOOCV MSE	h	γ
Weighted Kernel Regression	35930.0	2079.6	12.5	-
Local Linear Regression	22308.79	917.75	400	-
Local Linear Semi-supervised Regression	22869	902.107280	400	1

Table 4.2: Performance on the housing data set

4.6 Conclusion

We introduce Local Linear Semi-supervised Regression and show that it can be effective in taking advantage of unlabeled data. In particular, LLSR seems to perform somewhat better than WKR and LLR at fitting "peaks" and "valleys" where there are gaps in the labeled data. In general if the gaps between labeled data are not too big and the true function is "smooth" LLSR seems to achieve a lower true Mean Squared Error than the other algorithms.

Chapter 5

Completed Work: Practical Issues in Learning with Similarity Function

In this chapter we will describe the motivation for learning with similarity functions and give some experimental results on some synthetic examples.

5.1 Motivation

Kernels have become an extremely popular technique in the machine learning community. This popularity is largely due to the so-called "kernel trick" which allows kernels to work in high dimensional spaces without incurring a corresponding computational cost. Hence if data is not linearly separable in the original feature space kernel methods may be able to find a linear separator in some high dimensional space without too much computational cost.

However, in spite of the rich theory and practical applications of kernel methods, there are a few unsatisfactory aspects. In machine learning applications the intuition behind a kernel is that they serve as a measure of similarity between two objects. However, the theory of kernel methods talks about finding linear separators in high dimensional spaces that we may not even be able to calculate much less understand. This disconnect between the theory and practical applications makes it difficult to gain theoretical guidance in choosing good kernels for particular problems.

Secondly and perhaps more importantly, kernels are required to be symmetric and positive-semidefinite. The second condition in particular is not satisfied by many practically useful similarity functions. Hence if these similarity functions are to be used with kernel methods, they have to be coerced into a "legal" kernel. Such coercion may very reduce the quality of the similarity functions.

From such motivations, Balcan and Blum recently initiated the study of general similarity functions. Their theory gives a definition of a similarity function that has standard kernels as a special case and they show how it is possible to learn a linear separator with a similarity function and give similar guarantees to those that are obtained with kernel methods.

One interesting aspect of their work is that they give a prominent role to unlabeled data. In particular unlabeled data is used in defining the mapping that projects the data into a linearly separable space. This makes their technique very practical since unlabeled data is usually available in greater quantities than labeled data in most applications.

The work of Balcan and Blum provides a solid theoretical foundation, but its practical implications have not yet been fully explored. Practical algorithms for learning with similarity functions could be useful in a wide variety of areas, two prominent examples being bioinformatics and text learning. Considerable effort has been expended in developing specialized kernels for these domains. But in both cases, it is easy to define similarity functions that are not legal kernels but match well with our desired notions of similarity.

Hence, we propose to pursue a practical study of learning with similarity functions. In particular we are interested in understanding the conditions under which similarity functions can be practically useful and developing techniques to get the best performance when using similarity functions.

5.2 Background

5.2.1 Linear Separators and Large Margins

Machine learning algorithms based on linear separators attempt to find a hyperplane that separates the positive from the negative examples. i.e if example x has label $y \in \{+1, -1\}$ we want to find a vector w such that $y(w \cdot x) > 0$

Linear separators are currently among the most popular machine learning algorithms, both among practitioners and researchers. They have a rich theory and have been shown to be effective in many applications. Examples of linear separator algorithms are perceptron, winnow and SVM.

An important concept in linear separator algorithms is the notion of "margin." Margin is considered a property of the data set and (roughly speaking) represents the "gap" between the positive and negative examples. Theoretical analysis has shown that the performance of a linear separator algorithm is inversely proportional to the size of the margin.

5.2.2 The Kernel Trick

A kernel is a function $K(x, y)$ which satisfies certain conditions:

1. continuous
2. symmetric

3. positive semi-definite

If these conditions are satisfied then Mercer's theorem states that $K(x, y)$ can be expressed as a dot product in a high-dimensional space i.e there exists a function $\Phi(x)$ such that

$$K(x, y) = \Phi(x) \cdot \Phi(y)$$

Hence the function $\Phi(x)$ is an explicit mapping from the original space into a new possibly much higher dimensional space.

The "kernel trick" is essentially the fact that we can get the results of this high dimensional inner product without having to explicitly construct the mapping $\Phi(x)$.

5.2.3 Kernels and the Johnson-Lindenstrauss Lemma

The Johnson-Lindenstrauss Lemma states that a set of n points in a high dimensional Euclidean space can be mapped down into an $O(\log n/\epsilon^2)$ dimensional Euclidean space such that the distance between any two points changes by only a factor of $(1 \pm \epsilon)$.

Hence, if the data set has margin γ then we can map it into a space of dimension $\tilde{O}(1/\gamma^2)$. A kernel can be viewed as (implicitly) mapping the data into a high-dimensional space.

An interesting question is whether given black-box access to a kernel it is possible to map the data into a $\tilde{O}(1/\gamma^2)$-dimensional space such that if the data is separable with margin γ with the kernel it is also approximately separable in the new space. In other words we want to use the kernel to project the data into a lower-dimensional space where it is linearly separable.

Balcan, Blum and Vempala [1] answer this question in the affirmative by giving an explicit algorithm for performing such a mapping. An important point to note is that their algorithm requires access to the distribution where the examples come from in the form of unlabeled data. In the end, instead of having the linear separator live in some possibly infinite dimensional space we can compute a separator in a space whose dimension depends on the margin in the high-dimensional space.

5.2.4 A Theory of Learning With Similarity Functions

The mapping discussed in the previous section depended on K(x,y) being a legal kernel function. In [1] Balcan and Blum show that it is possible to use a similarity function which is not necessarily a legal kernel in a similar way to explicitly map the data into a new space. This mapping also makes use of unlabeled data.

Furthermore, similar guarantees hold: If the data was separable by the similarity function with a certain margin then it will be linearly separable in the new space. The implication is that any valid similarity function can be used to map the data into a new space and then a standard linear separator algorithm can be used for learning.

5.2.5 Winnow

We are particularly interested in studying learning with the Winnow algorithm. There are two main reasons for this.

1. Our approaches are based on augmenting the features of examples with a plethora of extra features. Winnow is known to be effective in dealing with many irrelevant features.
2. Experience indicates that unlabeled data becomes particularly useful in large quantities. In order to deal with large quantities of data we will need fast algorithms, winnow is a very fast algorithm and does not require a large amount of memory.

5.3 Learning with Similarity Functions

Suppose $\mathbb{K}(x, y)$ is our similarity function and the examples have dimension k

We will create the mapping $\Phi(x) : \mathbb{R}^k \rightarrow \mathbb{R}^d$ in the following manner:

1. Draw d examples $\{x_1, x_2, \ldots, x_d\}$ uniformly at random from the data set.
2. For each example x compute the mapping $x \rightarrow \{\mathbb{K}(x, x_1), \mathbb{K}(x, x_2), \ldots, \mathbb{K}(x, x_d)\}$

Although the mapping is very simple, in the next section we will see that it can be quite effective in practice.

5.4 Experimental Results on Synthetic Data Sets

To gain a better understanding of the algorithm we first performed some experiments on synthetic data sets.

5.4.1 Synthetic Data Set:Circle

The first data set we consider is a circle as shown in Figure 5.1 Clearly this data set is not linearly separable. The interesting question is whether we can use our mapping to map it into a linearly separable space.

We trained it on the original features and on the induced features. This experiment had 1000 examples and we averaged over 100 runs. Error bars correspond to 1 standard deviation. The results are see in 5.2. The similarity function that we used in this experiment is $\mathbb{K}(x, y) = (1/(1 + ||x - y||)$

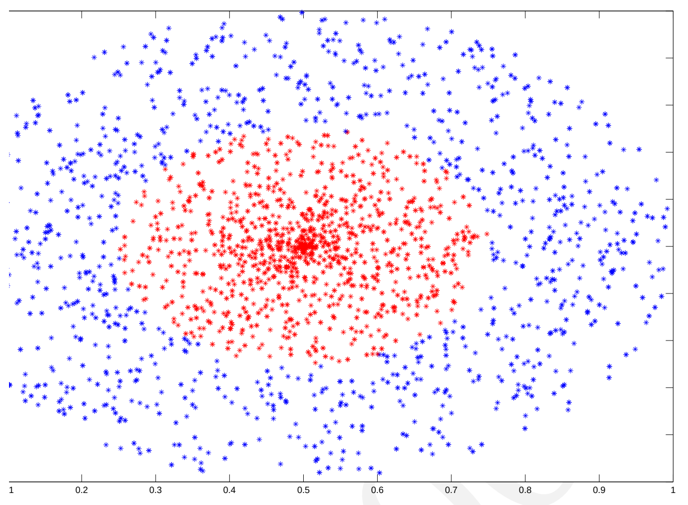

Figure 5.1: The Circle Data set

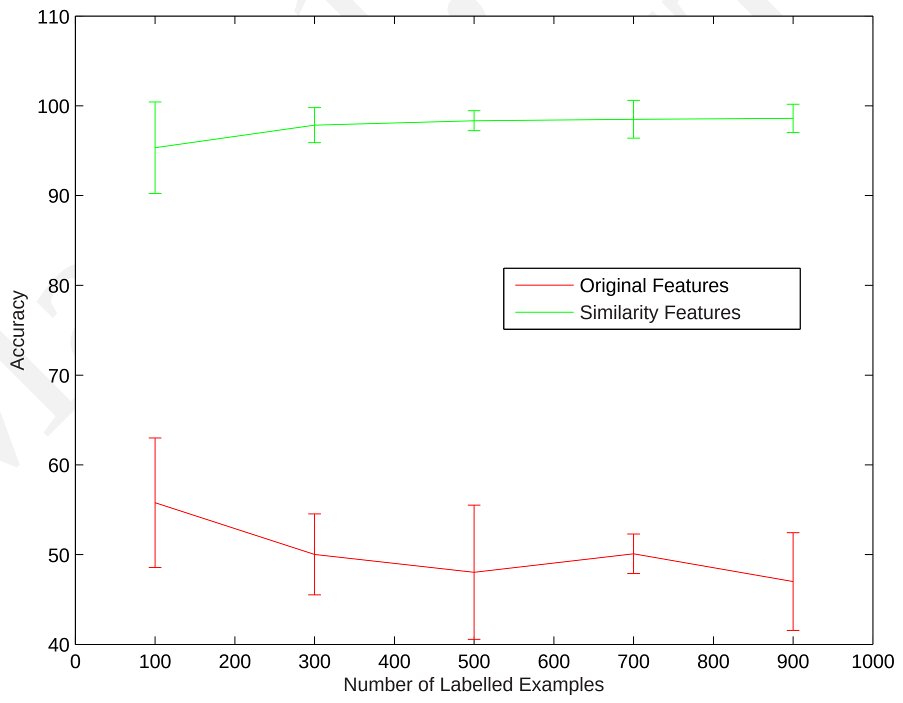

Figure 5.2: Accuracy vs training data on the Circle data set

In this case, clearly the induced features perform much better than the original features which are barely better than random.

5.4.2 Synthetic Data Set:Blobs and Line

We expect the original features to do well if the features are linearly separable and the similarity induced features to do particularly well if the data is clustered in "blob." One interesting question is whether we can construct a data set that needs BOTH the original features and the similarity features to do well.

We generated this data set in the following way:

1. We selected k point to be the centers of our blobs and randomly assign them labels in $-1, +1$
2. We flip a coin.
3. If it comes up heads then we set x to random boolean vector of dimension d and $y = x_1$ (the first coordinate of x).
4. If it comes up tails then we pick one of the k centers and flip r bits and set x equal to that and y equal to the label of the center.

The idea is that the data will be of two types, 50in the original features and 50spaces by themselves should be able to represent the combination well, but the features combined should be able to work well.

As before we trained it on the original features and on the induced features. But this time we also combined the original and induced features and trained on that. This experiment had 1000 examples and we averaged over 100 runs. Error bars correspond to 1 standard deviation. The results are see in 5.3. The similarity function that we used in this experiment is $\mathbb{K}(x, y) = (1/(1 + ||x - y||)$

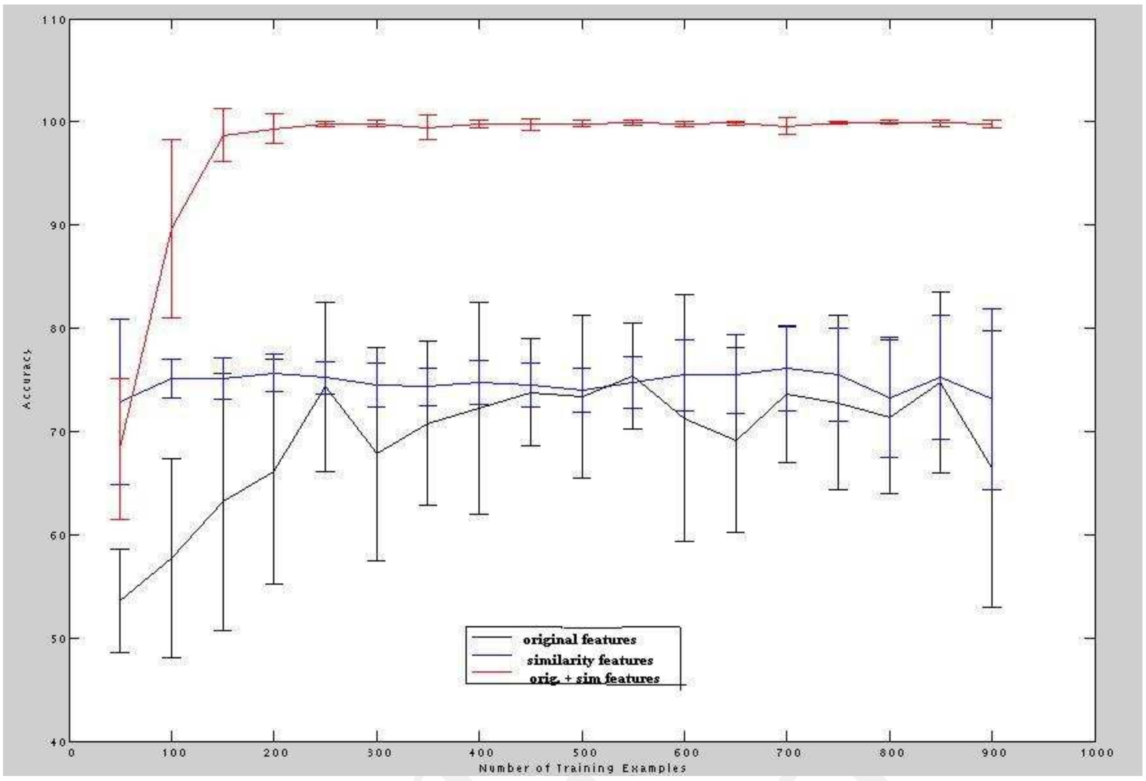

Figure 5.3: Accuracy vs training data on the Blobs and Line data set

As expected both the original features and the similarity features get about 75but the combined features are almost perfect in their classification accuracy. In particular this example shows that in at least some cases there may be advantages to augmenting the original features with additional features as opposed to just using the new features by themselves.

5.5 Conclusion

These results are encouraging and the potential for applications. Of course there is still much more to be done and in the next section we will describe the proposed experiments we plan to do to investigate the applicability of these techniques to real world data sets.

Chapter 6

Proposed Work

6.1 Local Linear Semi-supervised Regression

The completed experiments showed promising results. However, there are several issues that we would like to address:

6.1.1 Running Time

The naive formulation of the problem requires inverting a $O(nd \times nd)$ matrix where n is the number of examples and d is the dimension of the examples. Since a matrix inverse takes $O(n^3)$ time this becomes very impractical as n grows larger.

We intend to explore several ways to deal with this:

Sparsification

Operations on sparse matrices are typically faster than operations on dense matrices. One simple optimization would be to specify some ball of a certain radius around each point and ignore all point beyond this ball in the computations. The accuracy should not be affected by removing point which are far, but there might be significant savings in running time.

Iterative Methods for solving Linear Systems

There are several techniques for iteratively solving a system of linear equations $Ax = b$. They all work according to the same principle.

First we take the matrix A and split into two components such that

$A = M - N$

Hence,
$Ax = b$
$\Rightarrow (M - N)x = b$

$\Rightarrow x = M^{-1}(Nx + b)$

If we start off with some vector x^0 we can define a sequence $x^0, x^1, \ldots, x^k, x^{k+1}$ such that

$x^{(k+1)} = M^{-1}(Nx^k + b)$

It can be shown that if this sequence converges to x^* then x^* is the solution to $Ax = b$

The trick in applying this in choosing a matrix M that is easy to invert.

In the Jacobian methods we choose $M = D$ and $N = D - A$ where D is the diagonal matrix of A

In the Gauss-Seidel method, we choose M to be the lower triangular matrix of A and N to be the upper triangular part of A.

A sufficient (but not necessary) condition for both of these methods to converge is that A be "strictly diagonally dominant." This means that each diagonal entry should be greater than the sum of the remaining entries in its row.

While the combinatorial graph Laplacian satisfies this property, the Local Linear Laplacian satisfies a "block diagonally dominant" condition. Hence it is possible we can guarantee convergence if we treat each block as a unit.

Iterative methods are known to be particularly fast if the matrix is sparse. Hence this optimization technique can be combined with techniques for sparsifying the matrix.

Power Series

Recall that we arrive at an expression of the form
$\hat{y} = (Q + \gamma\Delta)^{-1}Py$
$\Rightarrow \hat{y} = (Q(I + \gamma Q^{-1}\Delta))^{-1}Py$
$\Rightarrow \hat{y} = (I + \gamma Q^{-1}\Delta)^{-1}Q^{-1}Py$
$\Rightarrow \hat{y} = (I - (-\gamma Q^{-1}\Delta))^{-1}Q^{-1}Py$

We can use the identity

$(I - A)^{-1} = I + A + A^2 + A^3 + \ldots$

In this case

$(I - (-\gamma Q^{-1}\Delta))^{-1} = I + (-\gamma Q^{-1}\Delta) + (-\gamma Q^{-1}\Delta)^2 + (-\gamma Q^{-1}\Delta)^3 + \ldots$

Hence,

$$\hat{y} = (I + (-\gamma Q^{-1}\Delta) + (-\gamma Q^{-1}\Delta)^2 + (-\gamma Q^{-1}\Delta)^3 + \ldots)Q^{-1}Py$$

$$\hat{y} = Q^{-1}Py + (-\gamma Q^{-1}\Delta)Q^{-1}Py + (-\gamma Q^{-1}\Delta)^2 Q^{-1}Py + (-\gamma Q^{-1}\Delta)^3 Q^{-1}Py + \ldots$$

Takin a few terms of the series may be a good approximation to the true value.

Doing supervised learning first

So far we have computed our estimate directly from first principle. But we haven't taken advantage of the fact that the semi-supervised result is usually very close to the supervised result. A better strategy would be to first compute the answer returned by a purely supervised method such as Local Linear Regression and then "smooth" this answer until we get to the semi-supervised result. Since the supervised answer can be obtained relatively fast, this could significantly speed up computation.

This idea could be implemented in conjunction with the iterative methods for solving a linear system by using the supervised solution as the starting point. Since this solution should be close to the final answer, this could significantly reduce the number of steps needed to converge.

6.1.2 Comparison with other semi-supervised regression algorithms

So far we have only compared against purely supervised methods such as Local Linear Regression. It would be interesting to compare our results against other algorithms for semi-supervised regression, particularly those which have a similar approach and to try and understand any reasons for significant differences in performance.

6.1.3 Experiment on a Wider Variety of Data sets

It is not clear to what extent the assumptions made by Local Linear Semi-supervised Regression are met by real world data sets. It would be interesting to experiment on a large enough variety of data sets so we can get an idea of which conditions allow LLSR to perform significantly better than comparable supervised methods or other semi-supervised regression methods.

6.2 Practical Learning with Similarity Functions

So far we have only started working on this project.

6.2.1 Protein Homology Detection

Protein structure prediction is a fundamental problem in biology. The problem is given the chemical formula for a protein to try and discover its 3-dimensional structure. It turns out that protein structure is an important determinant of the chemical properties of the protein. Hence, in applications such as drug design, if we want to predict the effect of a drug on a certain protein, we would be very interested in learning its structure.

Unfortunately, protein structure prediction, turns out to be a computationally difficult task. There are techniques which attempt to predict the structure of the protein from first principles, but so far these algorithms are too computationally intensive for practical purposes. So, the most popular technique is based on protein homology. What this means in simple terms is finding the protein sequence which is most similar to that of the given protein and predicting that the structure of the given protein will be similar.

However, for amino acid sequences, finding the most similar sequence is not a trivial thing. The most popular similarity measure is the Smith-Waterman score. This SW score is a measure of local alignment between two sequences and it can be computed optimally using local alignment. Empirically it has been shown to be the most useful measure of similarity between proteins.

However, as it stands the SW-score can not be used with kernel methods, because it does not satisfy they properties of a kernel. Hence, if we want to apply kernel methods to the protein homology problem, we will have to either:

1. Use other (probably inferior) measures of similarity.
2. Try to "coerce" the SW score into a legal kernel.

Both of these approaches have been tried in practice [60]. We propose a different approach. Since the SW score is a perfectly valid similarity function, we can use our techniques to extract a small number of features to use in a learning algorithm such as winnow. As a bonus, it happens that there are many unlabeled protein sequences available in this domain and our techniques allows us to use unlabeled data to our advantage.

Some of the issues we might be interested in:

1. Will using the similarity function directly prove more effective than the various kernelized measures?
2. What kind of modifications will be needed to the basic algorithm to make it perform better with biological sequences?

6.2.2 Using similarity in text learning

Text classification is one of the most active sub-fields in the machine learning community. It has numerous applications in information retrieval, web-search, natural language processing and so on. For the past 10 years the most popular technique has been Bag-Of-Words (BOW). In this technique Each document is transformed into a vector and learning is done on the vector that consists of the number of times each words appears in the document. Some adjustments are made to eliminate or reduce words that are extremely common such as "and" and "the."

But according to Gabrilovich and Markovitch [29], recently a plateau had been reached in using BOW. Not only does BOW not incorporate any information about the order of words in the document it does not allow for any domain knowledge to be incorporated. Gabrilovich et. al. propose to use domain knowledge by generating extra features for each document before feeding it into a text classifier. They generate these extra features using a hierarchical text classifier and freely available online encyclopedias.

By using this information, they are able to incorporate “general knowledge” into their features. For example the hierarchical classifier might see the word “Coke” and recognize that this could refer to the soft drink company, an illegal substance or a solid carbonaceous fuel and add a feature corresponding to the most likely meaning to the example. Using this technique Gabrilovich obtain significantly better performance than previous methods.

While the results of Gabrilovich et. al. ’s method is impressive it involves a significant amount of work in scouring the entire web to build a very comprehensive hierarchical text classifier. The technique we have outlined for learning with similarity has a similar approach in that we also add features that will hopefully be helpful to us in classification, but it is considerably easier to implement.

The most common used similarity measure in text classification is cosine similarity, which roughly measures the percentage of words in common that two documents have. Within the framework of the theory of learning with general similarity functions however, we have strong theoretical motivation for trying out more unusual kinds of similarity. As an example we might using the edit distance between two documents as a similarity function, this actually is similar to the Smith-Waterman local alignment score used in computational biology.

In the text classification domain, there is usually a large quantity of unlabeled data available. An interesting question is whether we can get significant improvements over existing methods by exploiting the unlabeled data. It would also be interesting to discover whether non-standard similarity functions perform better than the standard ones using our technique.

6.2.3 Domain specific Similarity Functions

This idea was implicitly stated in the previous two sections. Most machine learning algorithms design a very generic measure so similarity, such as Euclidean distance, or Hamming distance and apply it across many different domains. The results of Gabrilovich et. al. suggest that it may be more useful to design a specific similarity function for each data set. The SW score can be seen as one example of such a domain-specific similarity, but this approach could be extended further to other domains, where currently only very typical similarity functions are used.

An interesting question would be to investigate the principles for designing such similarity functions and under what conditions they give a significant boost in performance.

Chapter 7

Contributions and Timeline

7.1 Expected Contributions

The expected contributions of the final thesis are:

- Randomized Mincut, an extension of the graph mincut algorithm for semi-supervised learning.
- Experimental evaluation of graph mincut algorithm.
- Local Linear semi-supervised Regression(LLSR), a new algorithm for regression with labeled and unlabeled data
- Experimental evaluation of LLSR on real and synthetic data.
- Techniques for making learning with similarity functions work in practise.
- Experimental evaluation of such techniques in real world domains.

7.2 Timeline

Summer 2007 - Investigate methods for speeding up LLSR. Perform experiments on learning with similarities in the protein homology and text classification domain.

Fall 2007 - Compare LLSR with other semi-supervised regression algorithms. Explore the use of domain specific similarity functions.

Spring 2008 - Start writing thesis.

Summer 2008 - Finish writing thesis.

Bibliography

[1] M.-F. Balcan and A. Blum. On a theory of learning with similarity functions. *ICML06, 23rd International Conference on Machine Learning*, 2006. 1.2.3, 2.3, 5.2.3, 5.2.4

[2] M.-F. Balcan, A. Blum, and S. Vempala. Kernels as features: On kernels, margins and low-dimensional mappings. *ALT04, 15th International Conference on Algorithmic Learning Theory*, pages 194—205.

[3] R. Bekkerman, A. McCallum, and G. Huang. categorization of email into folders: Benchmark experiments on enron and sri corpora,. Technical Report IR-418, University of Massachusetts,, 2004. 2.3

[4] Mikhail Belkin, Partha Niyogi, and Vikas Sindhwani. Manifold regularization: A geometric framework for learning from labeled and unlabeled examples. *Journal of Machine Learning Research*, 7:2399–2434, 2006. 2.1.2

[5] G.M. Benedek and A. Itai. Learnability with respect to a fixed distribution. *Theoretical Computer Science*, 86:377—389, 1991. 3.4.2

[6] K. P. Bennett and A. Demiriz. Semi-supervised support vector machines. In *Advances in Neural Information Processing Systems 10*, pages 368—374. MIT Press, 1998.

[7] T. De Bie and N. Cristianini. Convex methods for transduction. In *Advances in Neural Information Processing Systems 16*, pages 73—80. MIT Press, 2004. 2.1.2

[8] T. De Bie and N. Cristianini. Convex transduction with the normalized cut. Technical Report 04-128, ESAT-SISTA, 2004. 2.1.2

[9] A. Blum. Empirical support for winnow and weighted majority algorithms: results on a calendar scheduling domain. *ICML*, 1995. 2.3

[10] A. Blum and S. Chawla. Learning from labeled and unlabeled data using graph mincuts. In *Proceedings of the 18th International Conference on Machine Learning*, pages 19—26. Morgan Kaufmann, 2001. (document), 1.1, 1.2.1, 1.2.2, 2.1.2, 3.1, 3.2, 3.5

[11] A. Blum and T. Mitchell. Combining labeled and unlabeled data with co-training. In *Proceedings of the 1998 Conference on Computational Learning Theory*, July 1998. 1.2.1

[12] A. Blum, J. Lafferty, M. Rwebangira, and R. Reddy. Semi-supervised learning using randomized mincuts. *ICML04, 21st International Conference on Machine Learning*, 2004. 2.1.2

[13] Yuri Boykov, Olga Veksler, and Ramin Zabih. Markov random fields with efficient approximations. In *IEEE Computer Vision and Pattern Recognition Conference*, June 1998.

2.1.2

[14] U. Brefeld, T. Gaertner, T. Scheffer, and S. Wrobel. Efficient co-regularized least squares regression. *ICML06, 23rd International Conference on Machine Learning*, 2006. 1.2.2, 2.2.3

[15] A. Broder, R. Krauthgamer, and M. Mitzenmacher. Improved classification via connectivity information. In *Symposium on Discrete Algorithms*, January 2000.

[16] J. I. Brown, Carl A. Hickman, Alan D. Sokal, and David G. Wagner. Chromatic roots of generalized theta graphs. *J. Combinatorial Theory, Series B*, 83:272—297, 2001. 3.3

[17] Vitor R. Carvalho and William W. Cohen. Notes on single-pass online learning. Technical Report CMU-LTI-06-002, Carnegie Mellon University, 2006. 2.3

[18] Vitor R. Carvalho and William W. Cohen. Single-pass online learning: Performance, voting schemes and online feature selection. In *Proceedings of International Conference on Knowledge Discovery and Data Mining (KDD 2006)*. 2.3

[19] V. Castelli and T.M. Cover. The relative value of labeled and unlabeled samples in pattern-recognition with an unknown mixing parameter. *IEEE Transactions on Information Theory*, 42(6):2102—2117, November 1996. 2.1.1

[20] C.Cortes and M.Mohri. On transductive regression. In *Advances in Neural Information Processing Systems 18*. MIT Press, 2006. (document), 1.2.2, 2.2.1

[21] S. Chakrabarty, B. Dom, and P. Indyk. Enhanced hypertext categorization using hyperlinks. In *Proceedings of ACM SIGMOD International Conference on Management of Data*, 1998.

[22] O. Chapelle, B. Schölkopf, and A. Zien, editors. *Semi-Supervised Learning*. MIT Press, Cambridge, MA, 2006. URL `http://www.kyb.tuebingen.mpg.de/ssl-book`. 2

[23] T. Cormen, C. Leiserson, and R. Rivest. *Introduction to Algorithms*. MIT Press, 1990.

[24] F.G. Cozman and I. Cohen. Unlabeled data can degrade classification performance of generative classifiers. In *Proceedings of the Fifteenth Florida Artificial Intelligence Research Society Conference*, pages 327—331, 2002. 2.1.1

[25] I. Dagan, Y. Karov, and D. Roth. Mistake driven learning in text categorization. In *EMNLP*, pages 55—63, 1997. 2.3

[26] A.P. Dempster, N.M. Laird, and D.B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society, Series B*, 39(1):1—38, 1977. 1.2.1, 2.1.1

[27] M. Dyer, L. A. Goldberg, C. Greenhill, and M. Jerrum. On the relative complexity of approximate counting problems. In *Proceedings of APPROX'00, Lecture Notes in Computer Science 1913*, pages 108—119, 2000. 3.2

[28] Y. Freund, Y. Mansour, and R.E. Schapire. Generalization bounds for averaged classifiers (how to be a Bayesian without believing). To appear in Annals of Statistics. Preliminary version appeared in Proceedings of the 8th International Workshop on Artificial Intelligence and Statistics, 2001, 2003. 3.4.2

[29] Evgeniy Gabrilovich and Shaul Markovitch. Feature generation for text categorization using world knowledge. In *Proceedings of the 19th International Joint Conference on Artificial Intelligence*, pages 1048—1053, Edinburgh, Scotand, August 2005. URL `http://www.cs.technion.ac.il/~gabr/papers/fg-tc-ijcai05.pdf`. 6.2.2

[30] D. Greig, B. Porteous, and A. Seheult. Exact maximum a posteriori estimation for binary images. *Journal of the Royal Statistical Society, Series B*, 51(2):271—279, 1989. 3.2

[31] Steve Hanneke. An analysis of graph cut size for transductive learning. In *the 23rd International Conference on Machine Learning*, 2006. 3.4.2

[32] Thomas Hofmann. Text categorization with labeled and unlabeled data: A generative model approach. In *NIPS 99 Workshop on Using Unlabeled Data for Supervised Learning*, 1999.

[33] J.J. Hull. A database for handwritten text recognition research. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16:550—554, 1994. 1.1, 3.6.1

[34] M. Jerrum and A. Sinclair. Polynomial-time approximation algorithms for the Ising model. *SIAM Journal on Computing*, 22:1087—1116, 1993. 3.2

[35] M. Jerrum and A. Sinclair. The Markov chain Monte Carlo method: An approach to approximate counting and integration. In D.S. Hochbaum, editor, *Approximation algorithms for* NP-*hard problems*. PWS Publishing, Boston, 1996.

[36] T. Joachims. Transductive learning via spectral graph partitioning. In *Proceedings of the* 20^{th} *International Conference on Machine Learning (ICML)*, pages 290—297, 2003. 2.1.2, 3.1, 3.4.2, 3.6

[37] T. Joachims. Transductive inference for text classification using support vector machines. In *Proceedings of the* 16^{th} *International Conference on Machine Learning (ICML)*, 1999. 2.1.2

[38] David Karger and Clifford Stein. A new approach to the minimum cut problem. *Journal of the ACM*, 43(4), 1996.

[39] J. Kleinberg. Detecting a network failure. In *Proc. 41st IEEE Symposium on Foundations of Computer Science*, pages 231—239, 2000. 3.4.1

[40] J. Kleinberg and E. Tardos. Approximation algorithms for classification problems with pairwise relationships: Metric labeling and markov random fields. In *40th Annual Symposium on Foundations of Computer Science*, 2000.

[41] J. Kleinberg, M. Sandler, and A. Slivkins. Network failure detection and graph connectivity. In *Proc. 15th ACM-SIAM Symposium on Discrete Algorithms*, pages 76—85, 2004. 3.4.1

[42] John Langford and John Shawe-Taylor. PAC-bayes and margins. In *Neural Information Processing Systems*, 2002. 3.4.2

[43] N. Littlestone. Learning quickly when irrelevant attributes abound: A new linear-threshold algorithm. *Machine Learning*, 1988. 2.3

[44] D. McAllester. PAC-bayesian stochastic model selection. *Machine Learning*, 51(1):5—21, 2003. 3.1, 3.4.2

[45] Tom M. Mitchell. *Machine Learning*. McGraw-Hill, 1997.

[46] K. Nigam, A. McCallum, S. Thrun, and T. Mitchell. Learning to classify text from labeled and unlabeled documents. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence*. AAAI Press, 1998. 2.1.1

[47] S. Della Pietra, V. Della Pietra, and J. Lafferty. Inducing features of random fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(4):380—393, April 1997.

[48] Joel Ratsaby and Santosh S. Venkatesh. Learning from a mixture of labeled and unlabeled examples with parametric side information. In *Proceedings of the 8th Annual Conference on Computational Learning Theory*, pages 412—417. ACM Press, New York, NY, 1995.

[49] S.T. Roweis and L.K. Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290:2323—2326, 2000.

[50] Sbastien Roy and Ingemar J. Cox. A maximum-flow formulation of the n-camera stereo correspondence problem. In *International Conference on Computer Vision (ICCV'98)*, pages 492—499, January 1998.

[51] J. Shi and J. Malik. Normalized cuts and image segmentation. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, pages 731—737, 1997.

[52] V. Sindhwani, P. Niyogi, and M. Belkin. A co-regularized approach to semi-supervised learning with multiple views. *Proc. of the 22nd ICML Workshop on Learning with Multiple Views*, 2005. (document), 1.2.1, 1.2.2, 2.2.3

[53] Dan Snow, Paul Viola, and Ramin Zabih. Exact voxel occupancy with graph cuts. In *IEEE Conference on Computer Vision and Pattern Recognition*, June 2000.

[54] Nathan Srebro. personal communication, 2007. 2.3

[55] Josh Tenenbaum, Vin de Silva, and John Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290, 2000.

[56] S. Thrun, T. Mitchell, and J. Cheng. The MONK's problems. a performance comparison of different learning algorithms. Technical Report CMU-CS-91-197, Carnegie Mellon University, December 1991.

[57] UCI. Repository of machine learning databases. http://www.ics.uci.edu/ mlearn/MLRepository.html, 2000.

[58] V. Vapnik, editor. *The Nature of Statistical Learning Theory*. Springer Verlag, New York, 1995. 2.1.2

[59] V. Vapnik, editor. *Statistical Learning Theory*. Wiley, New York, 1998. 2.1.2

[60] J.-P Vert, H. Saigo, and T. Akutsu. Local alignment kernels for biological sequences. In B. Schlkopf, K. Tsuda, and J.-P. Vert, editors, *Kernel methods in Computational Biology*, pages 131—154. MIT Press, Boston, 2004. 2.3, 6.2.1

[61] Z. Wu and R. Leahy. An optimal graph theoretic approach to data clustering: theory and its application to image segmentation. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 15:1101—1113, 1993.

[62] T. Zhang and F. J. Oles. A probability analysis on the value of unlabeled data for classification problems. In *Seventeenth International Conference on Machine Learning*, June

2000.

[63] Z.-H. Zhou and M. Li. Semi-supervised regression with co-training. *International Join Conference on Artificial Intelligence(IJCAI)*, 2005. (document), 1.2.2, 2.2.2

[64] X. Zhu. Semi-supervised learning literature survey. Technical Report 1530, Computer Sciences, University of Wisconsin-Madison, 2005. http://www.cs.wisc.edu/∼jerryzhu/pub/ssl_survey.pdf. 2

[65] X. Zhu. Semi-supervised learning with graphs. 2005. Doctoral Dissertation.

[66] X. Zhu, Z. Ghahramani, and J. Lafferty. Semi-supervised learning using Gaussian fields and harmonic functions. In *Proceedings of the 20th International Conference on Machine Learning*, pages 912—919, 2003. 1.1, 1.2.1, 1.2.2, 2.1.2, 2.1.2, 3.1, 3.4.2, 3.6, 3.6.1, 3.7, 4.2