LEARNING WITH CAPACITY CONTROL: A SEMI-SUPERVISED APPROACH

By

Ayhan Demiriz

A Thesis Submitted to the Graduate

Faculty of Rensselaer Polytechnic Institute

in Partial Fulfillment of the

Requirements for the Degree of

DOCTOR OF PHILOSOPHY

Major Subject: Decision Sciences and Engineering Systems

Approved by the
Examining Committee:

Kristin P. Bennett, Thesis Adviser

Mark J. Embrechts, Member

John E. Mitchell, Member

Nong Shang, Member

Mohammed J. Zaki, Member

Rensselaer Polytechnic Institute
Troy, New York

June 2000

LEARNING WITH CAPACITY CONTROL: A SEMI-SUPERVISED APPROACH

By

Ayhan Demiriz

An Abstract of a Thesis Submitted to the Graduate

Faculty of Rensselaer Polytechnic Institute

in Partial Fulfillment of the

Requirements for the Degree of

DOCTOR OF PHILOSOPHY

Major Subject: Decision Sciences and Engineering Systems

The original of the complete thesis is on file
in the Rensselaer Polytechnic Institute Library

Rensselaer Polytechnic Institute
Troy, New York

June 2000

CONTENTS

LIST OF TABLES

LIST OF FIGURES

ACKNOWLEDGMENT

In the name of Allah, Most Gracious, Most Merciful. Praise be to Allah, the Cherisher and Sustainer of the worlds; Most Gracious, Most Merciful; Master of the Day of Judgment. Thee do we worship, and Thine aid we seek. Show us the straight way, The way of those on whom Thou hast bestowed Thy Grace, those whose (portion) is not wrath, and who go not astray. *The Holy Qur'an, Surah 1 Al-Fatiha (The Opening).*

Read! in the name of thy Lord and Cherisher, Who created- Created man, out of a (mere) clot of congealed blood: Read! And thy Lord is Most Bountiful, He Who taught (the use of) the pen, Taught man that which he knew not. Day, but man doth transgress all bounds, In that he looketh upon himself as self-sufficient. Verily, to thy Lord is the return (of all). *The Holy Qur'an, Surah 96 (Verses 1-8) Iqraa (Read!).*

I have had many good moments during my graduate study in this city of Troy. However I ask people whom I might have caused trouble(s) to forgive me. It was not an easy 6 years at all. We must admit that one can not easily finish her/his Ph.D. without isolating herself/himself from other people at some level. Yes, I did that like the many other RPI (recently "Rensselaer") students - unfortunately.

I would like to first thank to my advisor Prof. Kristin P. Bennett for supporting and encouraging me during my Ph.D. study. She has always had brilliant ideas and eagerness to pursue them. I also would like to extend my thanks to the members of my dissertation committee for their constructive comments and ideas. I especially thank to Prof. M. J. Embrechts, since I had chance to expose myself to Data Mining in his class back in 1996.

This work could not be finished without help from Academic Computing Services at VCC. I believe that Mark D. Miller is a person whom I should remember all my life. His efforts and prompt responses helped me to solve many computational problems.

I also would like to mention about my friends from all colors and genders. I will not name them though. I thank to The Muslim Community of Troy especially Imam Djafer Sebkhaoui for helping me to keep myself spiritually sound.

This work was partially supported by NSF IRI-9702306, NSF IIS-9979860 and NSF DMS-9872019.

Finally, I give respect and love to my family. Although they were thousands of miles away from me, I always felt their presence in my heart. Yes, I am married too. Here is the place for you Honey. I love you very much!

ABSTRACT

A very basic problem in machine learning is the overfitting of empirical data. This problem occurs where learning processes construct overly strong models to explain the dependencies within empirical data. Often these strong models fail to perform well on the unseen data due to the strong bias towards the empirical data used in the learning task.

In order to prevent this overfitting problem, learning algorithms use different remedies. For example regularization and early stopping in neural networks help to yield better models that perform well on unseen data. Pruning is used in decision trees. Margin maximization techniques are used in Support Vector Machines. Such solutions, in general, are examples of capacity control techniques. By limiting the richness and the flexibility of the learning method, we expect to prevent overfitting problem.

In this research, we introduce learning methods with new ways of handling capacity control. These methods are used in supervised, unsupervised and semi-supervised learning approaches by incorporating all the available information on hand. Semi-supervised learning can exploit both labeled and unlabeled data. We first propose a genetic algorithm based model that uses unsupervised learning and unlabeled data (semi-supervised learning) as a new form of capacity control. We then use unlabeled data to influence margin maximization in support vector machines as an alternative form of capacity control based on all available information. Finally, we use capacity control in the label space for a boosting approach which combines the outputs of the many weak learning models by linear weighting. In general, we propose semi-supervised learning models based on both labeled and unlabeled data as new ways of capacity control. Methods proposed in this research have shown strong results on benchmark problems compared to alternative approaches.

CHAPTER 1
INTRODUCTION

Often, supervised learning problems can be posed as finding dependencies (functions) between the empirical input data and the outcome of the supervisor or supervision process. Three important issues must be addressed carefully in a such learning process (adapted from [98]):

1. To estimate a function from a large set of functions.
2. To estimate such function using very limited number of examples (empirical data).
3. To generalize well on unseen data .

Using overly strong or complex functions to define such dependencies can result in overfitting of the data and failure to generalize. For any learning task, limiting the richness and the flexibility of the class of functions being searched is called capacity control. By controlling the capacity, we can practically avoid the overfitting problem on empirical data.

The classical statistical approaches such as Maximum Likelihood (ML) method do not solve issues listed above. Therefore many methods such as Neural Networks (NN) and decision trees have been developed in learning framework to address these issues since 1960's. Classical approach uses ML for density estimation, discriminant analysis, and regression models. Since ML might fail even in the simple cases such as estimating the mixture of normal densities, classical parametric models might perform poorly in the learning process as measured by generalization on unseen data. On the other hand, non-parametric methods such as Parzen windows are considered to be more desirable in terms of estimating a density from a wide classes of densities compared to the parametric methods and have better *asymptotic* rate of convergence especially for smooth densities. Although they have remarkable asymptotic properties, experimental studies have shown that non-parametric methods did not perform significantly better than parametric methods when applied to very limited number of data points [98].

There are many forms of capacity control, for example regularization and early stopping in NN. In Bayesian inference, capacity control is performed via strong *a priori*. Specifically, Bayesian approaches perform well, if the function being estimated matches with *a priori* class of the functions provided in the Bayesian inference. In addition, *a priori* probability distribution of these functions should reflect the reality. Incomplete representation of the reality might slow down the convergence speed of Bayesian inference. Thus, Bayesian approach requires strong *a priori* to control the capacity of the learning process.

Supervised learning methods, in general, minimize a loss function based on the discrepancy between results from the model and the response from the supervisor. Minimizing this loss function just based on the empirical data can result in overfitting. Structural Risk Minimization (SRM)was introduced by statistical learning theory to find a trade-off between empirical risk and functional complexity [98]. Practically, SRM implements capacity control through a complexity penalty term [41]. In SRM, a structure is given as weak *a priori* to the learning process. SRM then finds optimal parameters for such a structure. For example in classification, according to the principles of SRM, for a fixed empirical risk level, finding the largest margin of separation that exists within the data prevents overfitting. In contrast, Bayesian inference has an implicit capacity control depending heavily on the strong *a priori* probability function provided by the user thus requires human interface in learning process.

In this research, different ways of capacity control are proposed based on incorporating all the available information on hand. For example if we are performing a classification task based on labeled training data, we would like to exploit any additional unlabeled data. The proposed methods perform capacity control on supervised and "semi-supervised" learning.

Typically learning tasks are divided into two categories: supervised learning and unsupervised learning. In supervised learning, training is done using data provided with the labels (e.g. dependent variable) and the resulting classifier is used to classify unlabeled data (e.g. data with unknown dependent variable). In this case, the expert needs to define classes explicitly and needs to label data prior to the training phase. The results are usually compared by ranking the generalization ability and accuracy on both labeled and unlabeled datasets.

Learning without known labels is known as unsupervised learning. Clustering is an important example of unsupervised learning methods. Clustering is simply defined as grouping similar objects. In clustering, classes or groups of the objects are not known *a priori* but rather an emergent property of the data. Clustering algorithms are valuable for discovering patterns in data. In many domains, the difficulty of labeling data and the lack of prior knowledge about classes sometimes limit us to use clustering algorithms to analyze and to group the data. For example, in order to classify a gene, a biochemist might have to run very expensive and time consuming experiments. By conducting cluster analysis, similar genes can be grouped together and results can be summarized to discover hidden relationships. Thus, useful information can be gathered from clustering even in the case of lack of information about the domain itself. A number of clustering algorithms are summarized in [58]. But most of them suffer from the following problems:(1) Choosing and validating the number of clusters and (2) Ensuring that the algorithmic findings represent the reality. Moreover,as stated in [13], k-means like clustering algorithms also suffer from the problem of evenly distributing the points among the clusters, since a sum of squares type of objective function is minimized. To form necessary foundation for supervised and unsupervised learning, we give a brief literature review in Chapter 2.

Depending on application domain, both supervised and unsupervised learning methods have superior properties compared to the other one. They also suffer from some weaknesses. For example, the validation of clusters found in cluster analysis could be an ill-defined process. In supervised learning, the difficulty in interpreting the classification results (unless we use a decision-tree like method) may make results less useful.

To avoid these weaknesses and to use superior properties, a semi-supervised approach is proposed in this research. Semi-supervised learning is defined as combining both labeled and unlabeled data to accomplish a learning task. By using a semi-supervised approach, we can control the capacity of a learning function based on all the information from the data. One possible way of doing is introducing partial supervision into unsupervised learning. Using the labeled data in semi-supervised learning will yield meaningful clusters and these clusters will be homogeneous. A semi-supervised clustering algorithm was introduced in [76]. It has been successfully applied to the segmentation of magnetic resonance images (MRI) as reported in [13] and [95]. Due to the difficulty of the labeling the data in the case of image segmentation, very limited number of points were labeled in [13]. A partially supervised clustering algorithm [13] is introduced to minimize a weighted sum of square objective function formed by using both the labeled and unlabeled data. An iterative scheme continues until no improvement is made on the objective more than a predetermined threshold. The prior weights and labels are determined by experts' knowledge. The authors reported better results by using semi-supervised fuzzy c-means algorithm compared to traditional fuzzy c-means algorithms on both artificial and MRI data [13]. Even in the case of very few labeled data, semi-supervised method improved results.

In Chapter 3, we introduce a new way of capacity control based on a semi-supervised clustering approach implemented with Genetic Algorithms (GA). Data are clustered using an unsupervised learning technique biased toward producing clusters as pure as possible in terms of class distribution. These clusters can then be used to predict the class of future points. One key additional benefit of this approach is that it allows unlabeled data with unknown class to be used to improve classification accuracy. The objective function of a traditional clustering technique, cluster dispersion in the k-means algorithm, is modified to minimize both the within cluster variance of the input attributes and a measure of cluster impurity based on the class labels. By contrast to k-means clustering algorithm, this implementation finds a near optimum number of clusters for a given parameter set at the end of the learning process. This means that optimum number of clusters is not a parameter in the model, however it is a product of the learning process. Minimizing the within cluster variance of the examples is a form of capacity control to prevent overfitting.

The GA method utilizes a parametric fitness function based on both cluster dispersion metric such as Davies-Bouldin Index (DBI) [36] and a class impurity measure such as the Gini index [24]. Having a parametric objective allows the user to solve classification, clustering and semi-supervised learning problems by using an appropriate parameter set. The experimental results show that using the class information often improves the generalization ability compared to unsupervised methods based only on the input attributes. Results from two different cluster dispersion measures, Mean Square Error (MSE) and Davies-Bouldin Index (DBI), are reported. Results show that using DBI in transductive model gives an advantage to improve the overall accuracy and the quality of the clusters found by the GA model. Benchmark studies also indicate that the method performs very well even when few training examples are available.

Training using information from unlabeled data might help to improve classification accuracy as well. Hence an alternative approach in semi-supervised learning is to use unlabeled data in supervised learning as proposed by Vapnik in the context of transduction [97]. The key task in transduction is to estimate the function values of the certain points instead of estimating function itself everywhere. This contrasts with induction, estimating function everywhere, since transduction estimates the labels in the working set which contains unlabeled data by implementing structural risk minimization on all the available information from both the training and the working sets. The capacity control is provided by margin maximization where the margin is measured on both the labeled and unlabeled data. Since we include the working set into our model and make use of it, one would expect to get better performance in terms of accuracy which is crucial for business decisions such as credit line issue, reviewing mortgage applications and other customer related financial services. On the other hand, by including extra points from unlabeled data, we are able to better understanding underlying input (population) distribution.

In Chapter 4, we introduce a semi-supervised support vector machine (S^3VM) method for capacity control. We use S^3VM to solve the overall risk minimization (ORM) problem posed by Vapnik. The ORM problem is to estimate the value of a classification function at the given points in the unlabeled working set. This contrasts with the standard learning problem of empirical risk minimization (ERM) which estimates the classification function at all the possible values. We propose a general S^{3}VM model that minimizes both the misclassification error and the function capacity based on all the available data. We show how the S^{3}VM model for 1-norm linear support vector machines can be converted to a mixed-integer program (S^{3}VM-MIP) and then solved exactly using integer programming. We implemented this (S^{3}VM-MIP) in AMPL and used CPLEX as an integer programming solver. Results of S^{3}VM-MIP and the standard ERM approaches are compared on eleven data sets. Our computational results support the statistical learning theory results on transduction showing that incorporating the working set data improves generalization when insufficient training information is available. In every case, S^{3}VM either improved or showed no significant difference in generalization compared to the ERM approach. This was the first known model to solve transduction problem succesfully in the literature [9].

In Chapter 4, we also propose two other variants of the S^{3}VM . The first approach is to use local learning to improve the convergence speed of the S^{3}VM. The advantages of using local learning models are very well discussed in [1]. The basic idea is to implement S^{3}VM in the k-nearest neighborhood of points which results in smaller integer-programming models to solve. We report improved results compared to S^{3}VM for the same datasets. The local learning approach is scalable to very large datasets. In the second approach, we investigate a gradient descent algorithm for the quadratic transduction problem. The numerical results based on proposed descent algorithm and the transductive algorithm based on SVM-Light [60] do not conclusively support the transductive approach. In general, algorithms for the quadratic transductive models are much slower and prone to local minima than those for the linear models. Our experimental study on these statistical learning methods indicates that incorporating the working set data for the capacity control can improve generalization ability, but the improvements were not large.

In Chapter 5, we again focus on capacity control in supervised learning but in a different context. In Chapter 4, the proposed algorithms perform capacity control in the feature space (input data). In Chapter 5, we specifically focus on capacity control in the label space using a boosting approach. The idea in boosting is to use a linear combination of many weak classification or regression functions (called weak learners), instead of one strong function. The resulting ensemble function frequently performs much better than any single function. Recent works by several people have shown boosting can be viewed as margin maximization in function space [87, 51]. Different boosting methods (e.g. AdaBoost [87]) can be viewed as gradient descent methods that minimize margin cost functions. In Chapter 5, we address the problem of the sensitivity of this function to outliers by adapting the soft margin cost function of support vector machines to boosting. Minimizing this soft margin error function directly optimizes a generalization error bound.

We formulate the problem as if all the possible weak learners had already been generated. The class labels produced by the weak learners become the new feature space of the problem. The boosting task becomes to construct a learning function in the label space that minimizes classification error and maximizes the soft margin. The resulting linear program can be efficiently solved using column generation techniques developed for large-scale optimization problems. The rows of the linear program each corresponding to one weak learner are generated as needed. The dual variables of the linear program provide the misclassification costs needed by the learning machine. The resulting "LPBoost" algorithm has many attractive properties. The algorithm has well defined convergence criteria. It converges in a finite number of iterations to a globally optimal solution. In computational experiments, the algorithm performs very well both in terms of computational time and generalization ability. The algorithm requires very few iterations. Thus few weak learners are actually generated and even fewer appear in the optimal learning ensemble. Numerical experiments are conducted by using both decision stumps and C4.5 [79]. The performance of the sparse learning ensembles produced was competitive with the other boosting approaches.

The basic idea in our research is to use all the available information in the learning task to improve the generalization ability through the capacity control whether extra information is in the form of raw data or outcomes from several models. We explore new types of the capacity control in the context of supervised and semi-supervised learning. The work here focuses on the classification problem where the labels are chosen from a finite set. In practice, the ideas presented in this research can be generalized to the regression problems where the data labels come from a continuous domain. Ideas for extensions to regression are summarized in the conclusion chapter.

CHAPTER 2
A Brief Literature Review on Data Analysis

2.1 Introduction

Historically statistics had a great impact on the traditional data analysis techniques. For example Fisher's Discriminant Analysis opened a new era in scientific research activities. Indeed scientific research depends on a solid and profound data and fact analysis. Classification has been used widely as a tool and representation technique in scientific research since ancient times. In scientific research, data frequently plays a central role. Depending on data an appropriate analysis technique may be used to come up with dependable conclusions. We will briefly summarize two major data analysis techniques in this chapter: cluster analysis based on unsupervised learning and classification based on supervised learning. Then, we look at a new area of data analysis - semi-supervised learning.

Cluster analysis is one of the very important unspervised learning methods. It is used in various research areas such as life, social and natural sciences. Thus, we can find different definitions for the terms cluster and cluster analysis in the literature based on various application areas. Indeed, scientists and researchers have given different definitions to the cluster analysis for the purpose of defining their own research problems in a proper manner. At the most general level, a cluster consists of similar objects collected or grouped together. Everitt documents [45] some of the definitions of a cluster:

- A cluster is a set of entities which are alike such that entities from different clusters are not alike.
- A cluster is an aggregation of the points in the test space such that the distance between any two points in the cluster is less than the distance between any two points in the cluster and any point not in it.
- Clusters may be described as connected regions of a multi-dimensional space containing a relatively high density of points.

According to Hansen and Jaumard [53], clusters are required to be homogeneous and/or well separated. Homogeneity means that entities within the same cluster should resemble one another and separation means that entities in different clusters should differ one from the other.

As indicated in [69], cluster analysis is mainly used for the data analysis as a multivariate statistical tool. It is also used for data summarization and compression. Storage and retrieval systems in hardware technologies also implement cluster analysis to improve access time to stored information. There are also many practical applications of cluster analysis in pattern recognition.

Besides cluster analysis as an example of unsupervised learning, we will give brief explanation of some classification techniques as a main method in supervised learning, particularly Support Vector Machines (SVM) to complete necessary background information. In their early work, Vapnik and Chervonenkis developed the optimal separating plane technique for classification and later on Vapnik introduced the idea of structural risk minimization which provides theoretical results to give the generalization error bounds of a separating hyperplane. The structural risk minimization and optimal separating hyperplane form the basis of SVM. Indeed, SVM is a logical extension to the optimal separating hyperplanes method. A SVM maps the input space into a high-dimensional feature space through some non-linear mapping (kernel functions) chosen a priori and then constructs the optimal separating hyperplane in the feature space. This mapping makes it possible to construct linear decision surfaces in the feature space which correspond to non-linear decision surfaces in the input space [93]. Having the ability to form non-linear decision surfaces in input space makes SVM a promising classification technique as a statistical learning method. A variation of SVM for data analysis is introduced in this study to do semi-supervised data analysis.

The outline of the rest of this chapter is follows. Section 2.2 focuses on cluster analysis, first, general approaches in cluster analysis are summarized. Later, examples of cluster analysis implementations are given in Section 2.2. We explain briefly some classification techniques including decision trees in Section 2.3 of this chapter. In Section 2.4, SVM method is explained in some details as an alternative learning method to the traditional ones. In Section 2.5, the semi-supervised learning problem is discussed in general and some related works to semi-supervised learning are reviewed.

2.2 Unsupervised Data Analysis Methods

As we mention above, a cluster is a group of similar objects based on a metric. Similarity or homogeneity is an important measure for the objects in the same cluster. On the other hand, we can also define the metric used in the cluster analysis in a way to minimize dissimilarity (dispersion) within the clusters. Like any other data analysis method, cluster analysis require some steps to follow. In their paper [53], Hansen and Jaumard approach cluster analysis from a mathematical point of view. Steps of a cluster analysis in a common framework are given below.

- Sample: Select a sample S of n entities among which clusters to be found where each entity is a vector.
- Data: Observe or measure p characteristic of the entities of S. This yields a $n \times p$ data matrix X.
- Dissimilarities from the matrix X a $n \times n$ matrix $D = (d_{kl})$ of dissimilarities between entities where $d_{kl} \geq 0, d_{kk} = 0, d_{kl} = d_{lk}$.
- Constraints: Choose the type of clustering desired. Specify also further constraints on the clusters, if any.
- Criterion: Choose a criterion to express homogeneity and/or separation of clusters in the clustering to be found.
- Algorithm: Choose or design an algorithm for the problem defined in Constraints and Criterion items.
- Computation: Apply the chosen algorithm to matrix $D = (d_{kl})$.
- Interpretation: Apply formal or informal tests to select the best clustering. Describe clusters by their descriptive statistics.

Some remarks could be made about dissimilarities at this point. First, dissimilarities may be computed from the sources other than a matrix of measurements X. Second, for some methods only the order of dissimilarities matters. Third, cluster analysis is not the only method to study dissimilarities. Fourth, instead of computing dissimilarities, one can perform a different type of cluster analysis (direct clustering) which requires using the matrix X. The clusters found by direct clustering correspond to concepts. Recently, conceptual clustering has become a very active field of research [42, 84].

There are basically five major types of clustering techniques mentioned in [53]: subset, partition, packing, covering, and hierarchy. More emphasis is given on the hierarchical clustering algorithms from the mathematical programming point of view. There are two main types of hierarchical clustering. First one is the agglomerative and second one is the divisive hierarchical clustering algorithm. Some algorithms for partitioning type of clustering from the dynamic programming, graph theoretical, branch and bound and cutting plane approaches are also listed.

One of the most cited references in the cluster analysis literature is written by Jain and Dubes [58]. In this book, clustering is defined as an exclusive and unsupervised classification. Clustering is exclusive in a sense each object belongs to the only one cluster, overlapping is not allowed. It is unsupervised, because objects are not labeled prior to implementation. Unsupervised classification (clustering) branches into two types of classification: hierarchical and partitional. Several algorithms can be proposed to express the same exclusive, unsupervised classification (clustering):

- Agglomerative vs. divisive: An agglomerative hierarchical clustering places each object in its own cluster and gradually merges these atomic clusters into larger ones. Thus, it is a bottom-top algorithm. A divisive hierarchical clustering algorithm, in the contrast, starts with one cluster and then splits this one down further. Thus, it is a top-down algorithm.

- Serial vs. simultaneous: Serial procedures handle the objects (patterns) one by one in an on-line process. Simultaneous procedures handle all the objects together in a batch process.

- Monothetic vs. polythetic: A monothetic clustering algorithm uses the features (variables) one at a time.

- Graph theory vs. matrix algebra: Some algorithms are expressed in terms of the graph theory. Some might be constructed algebraically especially using the vector algebra.

Two most known algorithms from the graph theory are single-link and complete-link algorithms. Single-link clusters are characterized as maximally connected subgraphs, whereas complete-link clusters are cliques or maximally complete subgraphs.

In partitional clustering, K-means [58] like algorithms are highly used and very popular. This kind of algorithms has local convergence. Usually the objective is to minimize mean square error within clusters and to maximize it between clusters.

The most important step in cluster analysis is the interpretation of the results. One important point should be addressed carefully: the validity of clusters. Problem might occur if different cluster methods yield different clusters which is high likely. Which partition is the correct one? Another source of doubt in clustering results is the number of clusters found within data. How do we know that which number is the correct one? Since there is almost no way to know the correct answers to these two questions, heuristic methods have been developed. Both global and local heuristic measures are given in [57].

Cluster analysis has been applied successfully in various disciplines. Pattern recognition is the one of the major fields. In [25], Buhmann summarizes Expectation Maximization (EM) type algorithms for data clustering and data visualization purposes. EM algorithms are stochastic optimization algorithms. Buhmann documents two conceptual approaches to cluster analysis.

- Parameter estimation of the mixture models by parametric statistics.

- Vector quantization of a data set by combinatorial optimization.

Parametric statistics assumes that noisy data have been generated by an unknown number of qualitatively similar stochastic processes. Each process is characterized by a unimodal probability density which is modeled by a parameterized mixture model e.g. Gaussian mixtures.

Vector quantization aims at finding a partition of the data set based an optimization principle. Partitioning type of clustering, according to Buhmann, arises in two different forms depending on the data format.

- Central clustering of the vectorial data.
- Pairwise clustering of the proximity data (dissimilarities).

In [25], Buhmann introduces five EM type algorithms for centroid estimation, pairwise clustering, and Multi-Dimensional Scaling (MDS) by deterministic annealing, structure preserving MDS. All algorithms have two common steps:

- E-like step: The expectation value of the complete data log-likelihood is calculated and conditioned on the observed data and the parameter estimates. This yields the expected assignment of data to mixture components.
- M-like step: The likelihood maximization step estimates the mixture parameters, e.g. the centers and the variances of the Gaussians.

In [55], the authors expand the idea in [25] to active data selection for clustering. The authors propose EM-like iteration scheme with the E-step replaced by the clustering algorithm itself. They also propose a criterion for the active data selection. They use this scheme in the data query frame. They report that the clustering cost decreases when this criterion is implemented.

A rigorous mathematical framework of Deterministic Annealing (DA) and Mean Field Approximation (MFA) is presented in [56]. The authors use the result to develop algorithms for an unsupervised texture segmentation which is equivalent to pairwise clustering problem, once an appropriate homogeneity measure has been identified. They compare the results of these algorithms with the other well-known ones. The optimization method used combines advantages of simulated annealing with the efficiency of a deterministic procedure. It has been applied successfully to a variety of combinatorial optimization problems and computer vision tasks.

Simulated annealing, a stochastic optimization strategy, has become very popular in recent years to solve image processing tasks. The random search is modeled by an inhomogeneous discrete-time Markov-chain which stochastically samples the solution space. The major disadvantage in simulated annealing is that the stochastic techniques might be extremely slow. But on the other hand, a slow annealing process yields very high quality partitions. The key idea in DA is to calculate the expectations of some relevant parameters analytically. Authors extend the idea of DA to meanfield annealing and they propose an algorithm for texture segmentation by using MFA. Gibbs sampling plays an important role in both DA and MFA.

Fisher, in [46], introduces an incremental conceptual clustering algorithm. A conceptual clustering system accepts a set of object descriptions and produces a classification scheme over the observations. This is an unsupervised learning task. It uses an evaluation function to discover classes with good conceptual descriptions. Thus, it is a type of learning by observation (as opposed to learning from examples) and it is an important way of summarizing the data in understandable manner. COBWEB , the algorithm proposed in [46], is an incremental and hierarchical clustering algorithm. It yields understandable tree structures. Incremental learning helps reducing the cost of clustering while preserving the quality of the conceptual description.

Clustering problems have been studied and applied in fuzzy logic. Bezdek analyzed data supplied by NASA [15]. The data set was collected remotely from the astronauts during space missions. He implemented a c-means fuzzy clustering algorithm which resembles the k-means algorithms. Overlapping is allowed in fuzzy clustering approach by fuzzy membership function. Thus an object might belong to more than one clusters. Fuzzy c-means clustering algorithm has some limitations e.g. it works for a very limited number (≤ 10) of clusters. But he reports that the quality of clusters are noteworthy and the error rates are very low.

Recently, cluster analysis has attracted a lot of attention from the area of data mining. One task in data mining is the search for hidden patterns that may exist in large databases. One application of cluster analysis is to “mine” spatial data. Ng and Han have developed the CLARANS [73] based on a randomized search. Experimental results are promising. Spatial data mining helps to extract interesting features, to capture intrinsic relationship between spatial and non-spatial data, to present the data regularity concisely, and to reorganize spatial databases to accommodate data semantics in order to improve performance.

We briefly summarized cluster analysis and its applications above as an example of unsupervised learning. Our focus in this research also requires knowledge on supervised learning methods. We summarize some classification methods in the next section.

2.3 Supervised Data Analysis Methods

Classification methods have been used and studied very much in the machine learning literature as well as in the other applied sciences and engineering. Among these methods tree-like ones are very common. Thus, decision trees have received a lot of attention from different disciplines. One of the earliest and well-known algorithms is ID3 developed by Quinlan. In [77], Quinlan introduces ID3 which is a top-down decision tree algorithm. A top-down algorithm starts from the root and splits data until it reaches the termination nodes. ID3 performs a non-incremental learning from examples. It branches the tree using information gain as the splitting criteria. The information gain is defined by the information theory and algorithm checks whether it decreases or not at a given node by splitting further. ID3 has been used extensively, since it was introduced. Recently other ID3-like algorithms have been developed and applied successfully [78]. The aims in developing new algorithms are finding better decisions, having faster algorithms, and getting more accurate learning strategies. C4.5 [78] is a very successful decision tree method. One of the important features of C4.5 is to enable the usage of misclassification costs. This is especially very important for the boosting applications and also for the unbalanced datasets.

Decision trees have also been used for regression. A regression tree is a tree-structured regression approach which allows complex models to be constructed from several lower order models. The decision tree partitions the data and a regression model is constructed in each partition. These models can be kept low order and hence be more interpretable. Regression trees also have the advantage that the simple form of the fitted function in each terminal node permits easily the study of the statistical properties of the model. In [29], Chaudhuri, Lu, Loh, and Yang propose a generalized regression tree algorithm. This method simply blends a tree-structured nonparametric regression and an adaptive recursive partitioning with maximum likelihood estimation. Traditional regression trees such as CART [24] (developed by Breiman, Friedman, Ulshen, and Stone) partition the regressor space and constant models are constructed at the leaf nodes of the regression trees. In essence, since models are constant at the leaf nodes, decision tree regression can be interpreted as a histogram approximation of the response surface. Since classification is a special case of the regression, CART is also used mostly for the classification purposes.

A large number of decision tree techniques have been proposed by the many authors in literature. Due to the lack of space, we will mention briefly a few of them. Lazy Decision Trees [49] were proposed by Friedman, Kohavi, and Yun in order to overcome some problems faced by traditional decision trees. Top-down decision tree algorithms implement a greedy approach that attempts to find a small tree. Most of the selection measures are based on one level of lookahead. According to authors, two related problems with the representation structure are the replication and the fragmentation. The replication problem forces duplication of the subtrees in disjunctive concepts. The fragmentation problem causes partitioning the data into smaller fragments. In order to avoid fragmentation problem as much as possible, the authors chose a test criteria that is a binary split on a single value and have allowed algorithm to split on any feature value is not equal to the instance's value. This algorithm is slow compared to the top-down decision trees. The induction process in lazy decision trees is delayed until a test instance is given. Thus it is a query based method (a form of local learning).

In decision tree construction, splitting criteria plays an important role. Univariate splits are common approach in the decision tree construction. In [6], Bennett and Blue propose a multivariate split approach for a fixed decision tree structure. A noteworthy point in this article is that the decision tree structure is fixed as in neural networks. Typically in implementation of decision trees, during the training phase a maximal tree is grown. In the pruning phase, the tree size is reduced to an optimal level by cutting out unnecessary branches. The proposed method in [6] does not require to implement any pruning algorithm in it, since it is a fixed structure. Because of multivariate fashion, method finds better decision rules and performs better in terms of overall accuracy. Bennett and Blue suggest using different optimization and search algorithms to solve the mathematical programming model for constructing such a decision tree.

One of the approaches for solving classification problem is the nearest neighbor method which is a form of local learning. Determining a proper metric plays very important role in this type of algorithms. In practice Euclidian-like distances are used mostly. In [48], Friedman proposes a flexible metric and related two algorithms to solve classification problems. He gives a solid statistical background on classification problems. The curse of dimensionality, variable subset selection and recursive partitioning are discussed from the nearest neighbor method point of view, prior to developing algorithms in this article. The machete is a recursive partitioning algorithm which splits only the most relevant variable. This way is more like a winner-take-all situation. In the opposite, the scythe tries to solve bias problem created by splitting by the most relevant variable. Various types of the splitting criteria such as purity index are discussed in [48].

Although numerous number of publications exist in the field of clustering and classification, very limited number of papers were reviewed in this chapter. Support Vector Machines has become very popular recently as a new way of machine learning method. Following section focuses on the theory of SVM and then some applications of them are summarized briefly.

2.4 Support Vector Machines

The classification problem has been studied extensively since Fischer introduced the notion of linear discrimination in mid 30's. Later, in the 60's Rosenblatt introduced the perceptron as a new way of machine learning. Until back-propagation was discovered by Rumelhart, Hinton, and Williams [83] in mid 80's, the perceptron method could not get enough attention because of some theoretic limitations. Since the mid 80's neural networks have become very popular in the field of pattern recognition and machine learning. Neural networks are a modification of the perceptron. Since the perceptron constructs a linear decision function, NN implement piece-wise linear type decision functions. A new type of learning method was constructed by Vapnik and Cortes called The Support Vector Machines [32]. SVM method implements the following idea: It maps input space into some high dimensional feature space. Then it constructs a linear decision surface in this feature space which corresponds to a non-linear decision surface in the original input space.

Two problems arise in this scheme: one is conceptual and the other one is technical. The conceptual problem is how to find a separating hyperplane that will generalize well. The dimensionality of the feature space will be huge and some separating planes will not necessarily generalize well. The technical problem is how computationally to treat such a high dimensional feature space (e.g. if a polynomial degree of 4 or 5 mapping in 200 dimensional input space is used, SVM may need to construct a billion dimensional feature space). Part of the conceptual problem was solved in 1965 for linearly separable cases by the optimal hyperplanes (maximal margin) method. Cortes and Vapnik [32] generalize this for linearly inseparable cases. In the optimal hyperplane technique, it was shown that if training vectors are separated without error, the expectation of committing error for test vectors is bounded by the ratio between the expectation of the number of support vectors and the number of training vectors, also called VC dimension. The generalization ability of the learning machine depends on the capacity of the set of functions, particularly VC dimension of these functions rather than the dimensionality of the input space. Functions with low capacity will generalize well on the unseen data regardless of the dimensionality of the data [99]. When the capacity is too large, the training dataset cannot be modeled properly due to the underfitting problem. On the other hand, If the capacity is too small, underlying model will overfit the training dataset. Geometrically, we can explain this phenomenon by large margin classifiers. The larger capacity in the set of functions used will result in larger margin ("fat") classifiers. Using a set of functions with low capacity will yield "skinny" margins with poor generalization [34].

Having addressed a solution to the conceptual problem, in 1992 it was shown that the order of operations for constructing a decision function can be interchanged

[19]. The original method required a non-linear transformation of input vectors to a higher dimensional space followed by inner products with support vectors in the high dimensional space. By using convolutions of the inner product in a Hilbert space, the high dimensional mapping and inner product can be performed in a single operation. The resulting method is called a kernel based method. This type of transformation enables the method to construct the rich classes of decision surfaces. Cortes and Vapnik call this type of machine learning Support Vector Machines.

Specifically, the optimal hyperplane method relies on the transformation of the p-dimensional input vector, x_i, into N-dimensional feature vector through a choice of N-dimensional vector function, ϕ, where $\phi : R^p - > R^N$. An N dimensional separator w and a bias, b, is then constructed for the set of transformed vectors

$$\phi(x_i) = \phi_1(x_i), \phi_2(x_i), ..., \phi_N(x_i) \qquad i = 1...n \tag{2.1}$$

where n is the number of observations (input vectors).

Classification of the unknown vector, x, is done by checking the sign of the function

$$f(x) = w \cdot \phi(x) + b \tag{2.2}$$

According to the properties of soft margin classifier method, w can be written as

$$w = \sum_{i=1}^{n} y_i \alpha_i \phi(x_i) \tag{2.3}$$

where y_i is the label of i^{th} input vector, α_i is the lagrange multiplier for i^{th} input vector. The linearity of product implies then

$$f(x) = \sum_{i=1}^{n} y_i \alpha_i \phi(x) \phi(x_i) \ + \ b \tag{2.4}$$

The points x_i with $\alpha_i \ > \ 0$ are known as support vectors, since these are the only points that influence the solution. The core of the support vector machines is the representation of the inner product as a kernel function

$$\phi(u)\phi(v) \equiv K(u,v). \tag{2.5}$$

The above representation is explained by Hilbert-Schmidt Theory [96]: Any symmetric function $K(u,v)$ with $K(u,v) \in L_2$ can be expanded in the form

$$K(u,v) = \sum_{i=1}^{\infty} \lambda_i \alpha_i \phi_i(u)\phi_i(v) \tag{2.6}$$

where $\lambda_i \in R$ and ϕ_i are eigenvalues and eigenfunctions

$$\int K(u,v)\phi_i(u)du = \lambda_i \phi_i(v) \tag{2.7}$$

of the integral operator defined by the kernel, $K(u,v)$. The kernel function can be selected based on the problem domain, but there is no common rule for using the right kernel function. One of the most popular kernel functions is the polynomial classifier degree of d, $K(u,v) = (uv+1)^d$. SVM with radial basis function can be implemented by employing the following kernel function

$$K(u,v) = exp\{-\frac{\|u-v\|^2}{2\sigma^2}\}. \tag{2.8}$$

It is shown that support vector machines have the ability to generalize well. The SVM method has been applied recently to different classification problems successfully [99]. A comparison of SVM with Gaussian Kernels to Radial Basis Function classifiers is given in [91]. The authors compare a K-means Gaussian RBF network with an SVM and a hybrid method. The hybrid method finds center by using SVM and then implements Gaussian RBF network. The hybrid method performed best most of the time in their experiments. The similarities between SVMs and other linear models such as linear discriminant, linear perceptron and other linear models were reported in [52]. Basically, similarities exist between the objective functions and gradient descent algorithms were used to optimize such objective functions. These methods also show similarities in the way the duality and probabilistic interpretations of the scores were exploited [52].

SVM approach has been applied to constructing decision trees by Bennett and Blue. In [7], they compare SVM with Global Tree Optimization, a hybrid model of SVM and GTO and some well known decision tree methods such as C4.5. They report encouraging results for the GTO/SVM method. In most of the experiments, the GTO/SVM method performed the best or very close to the best.

The SVM method has not only been applied to classification problems. Indeed, It has also been adapted to a variety of problems such as non-linear principal component analysis, regression analysis and functional approximation [96]. The results from this research are promising. Since it is a new research area, there are still things need to be done. Kernel PCA and non-linear PCA in feature space are first introduced in [90, 89]. The central focus in kernel PCA is to find principal components in the higher dimensional feature space rather than to find in the input space (linear PCA). The challenging issue in this approach is to find eigenvalues of the kernelized covariance matrix. Algebraic decompositions are given in [5]. Although the method is a non-linear PCA, non-linear optimization is never used in any of the steps. Since the method does not look for the eigenvalues in the full space, it is computationally comparable with the linear PCA. Another advantage of kernel PCA is to find the principal components in a rich and high dimensional feature space. This allows including non-linear terms into the principal components [5].

Other implementations are provided by authors in various articles. The improving the accuracy and speed of SVM is studied by Burges and Schölkopf in [26]. Vapnik, Golowich and Smola construct mathematics behind the functional approximation and regression estimation by using SVM [101]. They also report some experimental results on simulated data. Support vector regression machines were studied by Drucker, Burges, Kaufman, Smola & Vapnik in [43]. There have been many other applications of SVM recently. Due to the space limitation, we mention some of these applications. We will introduce semi-supervised learning approach in the next section.

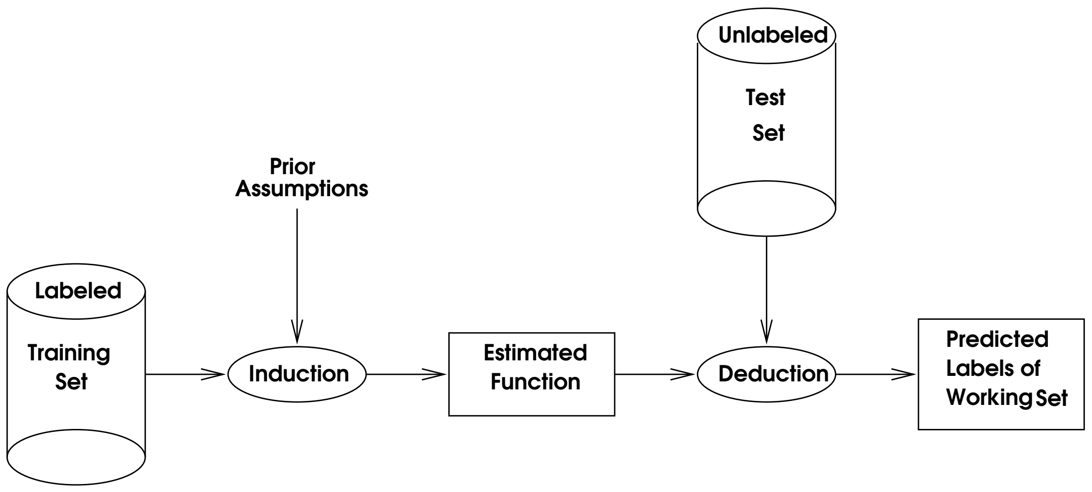


Figure 2.1: Inductive Learning

2.5 Semi-supervised Approach

Although the idea of incorporating unlabeled data into the learning process with the labeled data goes back to the early 80's [97], most of the applied techniques in machine learning deals with the labeled data in supervised learning. The unlabeled data is used in unsupervised learning. But easily available unlabeled data in many domains (e.g. web-based text data) [74, 67, 18] makes us to define a new type of learning. Because of this, combining labeled and unlabeled data in the learning process has gotten some attention from machine learning researchers.

As we stated in Chapter 1, we define semi-supervised learning problem as a use of a training set of the points with known classes and the working set of points without the class labels, constructing a classifier to label the working set. The idea of incorporating unlabeled data into supervised learning goes back to late 70s and early 80s [97, 100]. One version of semi-supervised learning is the transduction defined by Vapnik [99]. Unlike the inductive learning, transduction includes unlabeled data in a classification scheme in the learning phase. As it is depicted in Figure 2.1, inductive learning first, estimates a classifier function using labeled data (training set) and some prior knowledge about domain, and finally by using this estimated function, we classify unlabeled data in the deduction phase.

On the other hand, transduction does not require a classifier function estimation everywhere. Transduction involves combining both the labeled and unlabeled

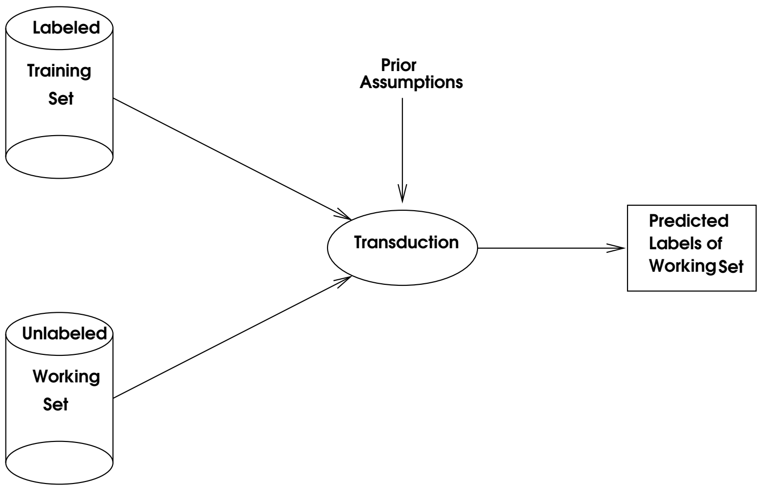


Figure 2.2: Transductive Learning

data with some prior knowledge to estimate the values of the classification function at working set points (unlabeled data) [99, 30], as depicted in Figure 2.2. This way of learning is very close to human learning because we often learn things by taking our experiences into consideration (labeled data) simultaneously with unknown subjects (unlabeled data) to explore and speculate on them. We propose a mixed-integer programming model for solving the transduction problem in Chapter 4.

Possible advantages of using both the labeled and unlabeled data led a group of researchers working on web-based text classification problem to develop an algorithm combination of the naive bayes and Expectation-Maximization (EM) algorithms [74, 67, 18]. The two-step algorithm is outlined in [74] as follows:

- Build an initial classifier by calculating classifier parameters from the labeled documents only (Naive-Bayes step).
- Loop while classifier parameters change.
  - Use the current classifier to calculate probabilistically weighted labels for the unlabeled documents (Expectation step).
  - Recalculate the classifier parameters given the probabilistically assigned labels (Maximization step).

The algorithm proposed in [74] is good particularly in the case of few labeled and many unlabeled data such as home page categorization, UseNet newsgroup message classification and Reuters news data. Essentially, labeling these types of data is very expensive despite the fact that unlabeled data is easily available. EM steps in this algorithm alternately generate probabilistically weighted labels for the unlabeled documents, and a more probable model with a smaller parameter variance. The key issue here is that using unlabeled data reduces the parameter variance. Variance reduction is due to the dependence between classification parameters and the random variable over the unlabeled data distribution.

EM based active learning methods are also considered as semi-supervised methods [67]. The Query-by-Committee (QBC) method of active learning has been implemented in [67]. Committee members are sampled from training data by the posterior Dirichlet distribution based on the training data word counts. This committee then approximates the entire classifier distribution. Traditional QBC then classifies unlabeled data and computes disagreement among the committee members for each label prediction. Finally for labeling requests, it documents the points on which members disagree most. The authors propose to measure committee disagreement for each document using Kullback-Leibler (KL) divergence to the mean criterion as opposed to previously employed vote entropy in which each committee member votes with probability mass $1/k$ for its winning class.

The KL divergence is an information-theoretic measure of the difference between two distributions, capturing the number of extra bits of information required to send messages sampled from the first distribution using a code that is optimal for the second. The difference between traditional QBC and this implementation is that an EM scheme with unlabeled data is performed after the step of finding the classifier function based on committee members and before computing the disagreement among the members for each unlabeled data point.

In [18], the authors combine labeled and unlabeled data through co-training using a Bayesian network type of classifier. Co-training first finds weak predictors by using the labeled data based on each kind of information (e.g. "research interests" might be a weak predictor for a faculty home page). Using the unlabeled data, the method generates a bootstrap sample from these weak predictors in the following step. Finally a Bayesian network classifier is implemented to classify this bootstrap sample. Authors reported some improvement compared to fully supervised training.

Another way of incorporating the unlabeled data into the learning process is to modify the objective function, which is optimized within a learning process. Çataltepe and Magdon-Ismail [28] proposed an augmented error, which has components from both the labeled and unlabeled data. The authors provide an analytical solution in the case of linear, noisy targets and linear hypothesis functions. They also show some results for the non-linear case.

Assume a training set: $\{(x_1, f_1), ..., (x_n, f_n)\}$ and the objective of the learning process is to choose a hypothesis g_v, among a class of hypotheses G, minimizing the test error over the test set of $\{(y_1, h_1), ..., (y_\ell, h_\ell)\}$. We can define training error in this case:

$$E_0(g_v) = \frac{1}{n}\sum_{i=1}^{n}(g_v(x_i) - f_i)^2$$

and test error as:

$$E(g_v) = \frac{1}{\ell}\sum_{j=1}^{\ell}(g_v(y_j) - h_j)^2.$$

Expanding the test error will give us:

$$E(g_v) = \frac{1}{\ell}\sum_{j=1}^{\ell} g_v^2(y_j) - \frac{2}{\ell}\sum_{j=1}^{\ell} g_v(y_j)h_j + \frac{1}{\ell}\sum_{j=1}^{\ell} h_j^2.$$

Since we can not speculate on the true labels of test set, Çataltepe and Magdon-Ismail then modify this error function as follows:

$$\begin{aligned} E(g_v) &\approx \frac{1}{\ell}\sum_{j=1}^{\ell} g_v^2(y_j) - \frac{2}{n}\sum_{i=1}^{n} g_v(x_i)f_i + \frac{1}{n}\sum_{i=1}^{n} f_i^2 \qquad (2.9) \\ &= E_0(g_v) + \frac{1}{\ell}\sum_{j=1}^{\ell} g_v^2(y_j) - \frac{1}{n}\sum_{i=1}^{n} f_i^2 \end{aligned}$$

By introducing an augmentation parameter α between 0 and 1, a more general error function, called the augmented error, can be defined as:

$$E_\alpha(g_v) \;=\; E_0(g_v) + \alpha(\frac{1}{\ell}\sum_{j=1}^{\ell} g_v^2(y_j) - \frac{1}{n}\sum_{i=1}^{n} f_i^2). \qquad (2.10)$$

For $\alpha = 0$, the augmented error is equivalent to training error E_0 and for $\alpha = 1$ corresponds to Equation 2.9.

We summarized related literature in clustering, classification and support vector machines in this chapter. We also reported some work done in the field of combining labeled and unlabeled data. We now examine the first semi-supervised method of this research in the next chapter. A GA based semi-supervised clustering method is proposed. From both GA and the semi-supervised learning perspectives, the proposed method has many innovative features.

CHAPTER 3
A Genetic Algorithm Approach for Semi-supervised Clustering

3.1 Introduction

In this chapter, incorporating label information into an unsupervised learning approach is studied. The goal is to group both labeled and unlabeled data into the clusters where each cluster is as pure as possible in terms of class distribution provided by the labeled data. The advantage of such an approach is that it can be used for both inductive and transductive inference. Moreover, unsupervised learning provides capacity control for classification. In addition the clusters help also characterize segments of the population likely or unlikely to possess the target characteristic represented by the label. This additional information can be useful for several applications. For example, in database marketing only the most pure clusters of customer would be included in a marketing campaign and new products may be designed to reach customers in marginal clusters. The work based on this chapter is also reported in [39, 38].

As a base to our semi-supervised algorithm, an unsupervised clustering method optimized with a genetic algorithm is used by incorporating a measure of classification accuracy used in decision tree algorithms, the GINI index [24]. Clustering algorithms that minimize some objective function applied to K-cluster centers are examined in this chapter. Each point is assigned to the nearest cluster center by Euclidean distance. The goal is to choose the cluster centers that minimize some measure of cluster quality. Typically a cluster dispersion metric is used. If the mean square error, a measure of within cluster variance, is used then the problem becomes similar to the classic K-means clustering [58]. An alternative metric, the Davies-Bouldin Index (DBI) [36] that is a function of both the within cluster variance and between cluster center distances is also examined. By minimizing an objective function that minimizes a linear combination of the cluster dispersion measure and the Gini Index, the algorithm becomes semi-supervised. The details of the problem formulation are given in Section 3.2. Since the objective function is highly nonlinear and discontinuous with many local minima, it is optimized by using the C++ based genetic algorithm library package GAlib [102].

Genetic Algorithms (GAs) are well known for being able to deal with complex search problems by implementing an evolutionary stochastic search. Because of this, GAs have been successfully applied to a variety of challenging optimization problems. The NP-hard nature of the clustering problem makes GA a natural choice for solving it such as in [13, 62, 85, 70, 35]. A common objective function in these implementations is to minimize the square error of the cluster dispersion:

$$E = \sum_{k=1}^{K} \sum_{x \in C_k} \|x - m_k\|^2 \tag{3.1}$$

where K is the number of clusters and the variable m_k is the center of cluster C_k. This is indeed the objective function for the K-means clustering algorithms. The algorithm proposed in [85] modifies this objective function by using the inverse of Davies-Bouldin index defined in [36, 58] and minimizing it by using evolutionary programming.

GAs are also implemented in [70] to minimize the function E (3.1). Genomes are represented by n-bit long strings where n is the number of data points. Each bit in genomes represents cluster membership. Although proper crossover and mutation operations are defined for this scheme, it is not a scalable algorithm due to the length of the genomes.

There are prior studies on cluster analysis using genetic algorithms such as an algorithm based on GAs is used for machine vision (pattern recognition) in [35]. The basic problem defined in [35] is to group objects to a fixed number of clusters. A new gene representation, Boolean Matching Code (BMC), is defined in [35] as an alternative way to Linear Code (LC) used in [70]. The basic idea in BMC is that each gene represents one cluster with n binary bits where n is the number of objects. In this case the total size of solution space is 2^{Kn}. Although the size of the solution space in LC is 2^{nlogK}, BMC reaches the convergence earlier than LC by utilizing better crossover and mutation operations. A single gene crossover operation was proposed in [35].

Since the proposed algorithm performs transduction using both labeled and unlabeled data in the learning task, there are some substantial differences between the proposed algorithm in this chapter and the algorithms proposed in [85, 70, 35]. But the core implementation of the GA has some similarities. Recently GA clustering was also implemented in the context of design and discovery of pharmaceuticals [44].

In Section 3.2, the problem definition and the proposed algorithm are given. Details about the GA implementation are given in Section 3.3. Experimental results are given in Section 3.4. A comparison with 3-nearest-neighbor and linear and quadratic discriminant analyses is also reported in Section 3.4. Finally, we summarize our findings in the Section 3.5.

Related approaches for combining supervised and unsupervised learning exist. For example Learning Vector Quantization (LVQ) [61] and Constrained Topological Maps (CTM) [31] also use the approach of adapting a primarily unsupervised method to perform classification. This chapter helps address the interesting, but still open question, of how well such methods can exploit the information in unlabeled data to support transductive inference. Moreover, nearest prototype classifiers are studied in [16, 62]. Selecting prototypes from dataset with GA is compared with random selection in [62].

3.2 Problem Definition

Clustering, in general, is defined as grouping similar objects together by optimizing some similarity measure for each cluster such as within group variance. Since clustering generally works in an unsupervised fashion, it is not necessarily guaranteed to group the same type (class) of objects together. In this case, supervision needs to be introduced to the learning scheme through some measure of cluster impurity. The basic idea is to find a set of clusters then minimize a linear combination of the cluster dispersion and cluster impurity measures. More specifically, select $K > 2$ cluster centers, m_k $(k = 1, ..., K)$, that minimize the following objective function:

$$\min_{m_k, k=1,\dots,K} \quad \beta * Cluster\_Dispersion \ + \ \alpha * Cluster\_Impurity \qquad (3.2)$$

where $\alpha > 0$ and $\beta > 0$ are positive regularization parameters.

If $\alpha = 0$, then the result is a purely unsupervised clustering algorithm. If $\beta = 0$ the result is a purely supervised algorithm that tries to minimize the cluster impurity. As in the K-means algorithm, each point is assumed to belong to the nearest cluster center as measured by Euclidean distance. Each non-empty cluster is assigned a class label corresponding to the majority class of points belonging to that cluster. Two cluster dispersion measures from the clustering literature will be examined: mean square error (see Section 3.2.1) and Davies-Bouldin Index (see Section 3.2.2). It is important to note that for the induction case, cluster dispersion is based on the labeled training data. For the transduction case, the cluster dispersion is based on all available data, both labeled and unlabeled. For the cluster impurity measure, a measure of partition quality common in decision tree algorithms, the Gini Index, is used (see Section 3.2.3). Since the objective function (Eq.3.2) is highly discontinuous with many local minima, it is optimized by using the genetic algorithm library (see Section 3.3) GAlib. When a solution is found, it might contain clusters with little or no points assigned to them. These clusters are deleted and relevant points are reassigned to their nearest cluster centers. Practical details of this algorithm are discussed in the computational results section (see Section 3.4). The resulting algorithm can be summarized as follows:

Algorithm 3.2.1. *Semi-supervised clustering algorithm*

- *Within genetic algorithm:*

    1. *Determine cluster centers*
    2. *Partition the labeled data by distance to closest cluster center.*
    3. *Find non-empty clusters, assign a label to non-empty clusters by majority class vote within them.*
    4. *Compute dispersion and impurity measures:*

  - *Induction: Use labeled data.*
  - *Transduction: Use labeled + unlabeled data.*

- *Prune clusters with few members.*
- *Reassign the points to final non-empty clusters.*

3.2.1 First Dispersion Measure: MSE

The average within cluster variance is frequently used in clustering techniques as a measure of cluster quality. Commonly known as the mean square error (MSE), this quantity is defined as:

$$MSE = \frac{1}{n}\sum_{k=1}^{K}\sum_{x\in C_k}\|x-m_k\|^2, \tag{3.3}$$

where n is the number of points, K is the number of clusters, and m_k is the center of cluster C_k. The K-means algorithm minimizes the MSE objective.

3.2.2 Second Dispersion Measure:DBI

The Davies-Bouldin index is used as an alternative to MSE. DBI is determined as follows [58] : Given a partition of the n points into K clusters, one first defines the following measure of within-to-between cluster spread for two clusters, C_j and C_k for $1 \leq j,k \leq K$ and $j \neq k$.

$$R_{j,k} = \frac{e_j + e_k}{D_{jk}}, \tag{3.4}$$

where e_j and e_k are the average dispersion of C_j and C_k, and D_{jk} is the Euclidean distance between C_j and C_k. If m_j and m_k are the centers of C_j and C_k, then

$$e_j = \frac{1}{n_j}\sum_{x\in C_j}\|x-m_j\|^2 \tag{3.5}$$

and $D_{jk} = \|m_j - m_k\|^2$, where m_j is the center of cluster C_j consisting of n_j points.

The term R_k for each C_k is defined as

$$R_k = max_{j \neq k} R_{j,k}. \tag{3.6}$$

Now the DBI is defined as:

$$DB(K) = 1/K \sum_{k=1}^{K} R_k \tag{3.7}$$

The DBI can be incorporated into any clustering algorithm to evaluate a particular segmentation of data. The DBI takes into account cluster dispersion and the distance between cluster means . Well separated compact clusters are preferred. The DBI favors small numbers of clusters. Optimizing the DBI frequently eliminates clusters by forcing them to be empty.

3.2.3 Impurity Measure: Gini-Index

The Gini index has been used extensively in the literature to determine the impurity of a certain split in decision trees [24]. Usually, the root and intermediate nodes are partitioned to two children nodes. In this case, left and right nodes will have different Gini index values. If any split reduces the impurity, the decision tree at that node is partitioned further and the decision rule which yields the minimum impurity is selected. Clustering using K cluster centers partitions the input space into K regions. Therefore clustering can be considered as a K-nary partition at a particular node in a decision tree, and the Gini index can be applied to determine the impurity of such a partition. In this case, Gini Index of a certain cluster is computed as:

$$GiniP_j \;=\; 1.0 - \sum_{i=1}^{c} (\frac{P_{ji}}{n_j})^2 \qquad j \; in \; 1, ..., K \tag{3.8}$$

where c is the number of classes and P_{ji} is the number of points belong to i^{th} class in cluster j. n_j is the total number of points in cluster j. The impurity measure of a particular partitioning into K clusters is:

$$impurity = \frac{\sum_{j=1}^{K} T_{P_j} * GiniP_j}{n} \tag{3.9}$$

where T_{P_j} is the probability of a point belonging to cluster j and n is the number of points in the dataset.

Preliminary experiments indicated that the Gini index is generally preferable over other impurity measures such as the number of misclassified points. If the simple number of misclassified points is used, then the misclassified points may be distributed evenly throughout all the clusters. The Gini Index favors solutions with pure clusters even at the expense of total classification error. Other decision tree splitting criterion such as Information Gain could also be used for the cluster impurity measure [78].

3.3 Genetic Representation and Algorithm

The objective function (Eq. 3.2) defined in the previous section is discontinuous and non-convex. Finding an optimal solution to this problem is extremely difficult, so heuristic search is desirable. Heuristic search approaches such as genetic algorithms (GA), evolutionary programming, simulated annealing and tabu search have been used extensively in the literature to optimize related problems. Genetic algorithm approach is utilized because the objective function defined can be readily used as a fitness function in the GA. The authors in [85, 70, 35] used their own customized GAs for clustering. As opposed to developing a genetic algorithm from scratch, a general purpose GA library, GAlib [102], is customized by utilizing the floating-point representation and Goldberg's simple GA approach [50]. This algorithm uses non-overlapping populations. In each generation, the algorithm creates an entirely new population of individuals by selecting from the previous population then mating to produce the new offspring for the new population. This process continues until stopping criteria have been met. An elitist strategy was applied which allows the best individual to pass to the new generation.

In a genetic algorithm application major concerns are genome representation, initialization, selection, crossover and mutation operators, stopping criteria and most importantly the fitness function. The objective function (Eq.3.2), defined above, is directly used as the fitness function without any scaling. The genome representation consists of an array of Kp real numbers, where p is the number of dimensions in the data and K is the number of clusters. Each set of p numbers represents one cluster center. This type of representation brings several advantages over prior discrete representation of cluster membership. First, cluster memberships are assigned based on Euclidean distance metric in this case instead of assigning them based on the values of genome. Second, each genome requires less search space than previous applications for the large datasets, since the length of the genomes depends only on the number of clusters (K) and the dimensionality (p) of the dataset, not the number of data points. It is therefore possible to handle the large datasets with this representation.

Default genetic operators defined for GARealGenome class in GAlib were applied. A mutation with Gaussian noise is the default in this case. Uniform crossover was applied as the default crossover operation. Although the uniform initializer is used by default, the population was initialized by sampling from the data. A uniform selection rule was used for selecting mating individuals (parents).

Two stopping criteria were applied . The algorithm stops when either of them is satisfied. One of these criteria is the maximum number of generations. The other one is the convergence after a certain number of consecutive generations.

The genetic algorithm yields reasonable results for both induction and transduction problems. The experimental findings are summarized in the next section.

3.4 Experimental Results

The goals in this computational approach are to determine if combining supervised and unsupervised learning approaches techniques could lead to improved generalization, and to investigate if performing transductive inference using unlabeled data for training could lead to improvements over inductive inference. Experimental study is done with eight datasets from the UC-Irvine Machine Learning

Repository [68][1]. The datasets have all originally two-class output variable except Housing. The output variable for this dataset was categorized at the level of 21.5. Each dataset was divided into three subsets after a standard normalization. These subsets are called the training, testing and working sets. Currently 40% of data is for training, 30% is for testing and remaining 30% is for working sets. For each dataset, two scenarios have been tested to understand the difference between inductive and transductive inferences. For inductive inference, the algorithm is applied to labeled training data and then tested on the test data. For transductive inference, the algorithm is applied to labeled training data, unlabeled working data, and unlabeled test data, and then tested on the test data.

Results from seven different fitness functions are reported. The two different cluster dispersion measures, MSE (Eq.3.3) and Davies-Bouldin Index (Eq.3.7), are applied to induction in a completely unsupervised mode ($\beta = 1, \alpha = 0$) and semi-supervised mode ($\alpha > 0, \beta > 0$), and transduction in a semi-supervised mode ($\alpha > 0, \beta > 0$). We also tried the completely supervised case based on only the Gini index ($\beta = 0, \alpha = 1$). For transduction, both the cluster dispersion measure and the Gini index are based on the labeled and unlabeled data. In transduction, the Gini index (Eq.3.8) becomes:

$$GiniP_j \;=\; 1.0 - \sum_{i=1}^{c} (\frac{P_{ji}}{\hat{n}_j})^2 \qquad j \; in \; 1, ..., K \tag{3.10}$$

$\hat{n}_j$ is equal to number of labeled and unlabeled points in cluster j.

The best parameter set for the problem was picked by trial and error. We use same set of GA parameters for each dataset. The maximum number of generations is 500, mutation probability is 0.01, probability of crossover is 0.95, and number of generations to converge is 50. Each generation consists of 50 competing genes. Experiments are conducted based on 10 bootstrap samples. For brevity only the average testing set error results are reported here. A paired t-test was used to assess the significance of difference of the testing set errors within a dataset. Errors

[1]The datasets and their corresponding sizes are: Bright(14 variables, 2462 points), Sonar(60,208), Cleveland Heart(13,297), Ionosphere(34,351), Boston Housing(13,506), House Votes (16,435), Breast Cancer Prognosis(30,569), and Pima Diabetes (8,769)

with a p-value less than 0.2 were considered significant. To insure that the weaker performance of MSE was not based on poor choice of parameters, (K, β, α) for each dataset were chosen based on trials with the inductive MSE with Gini index[2] For the DBI based results, the same values of K were used for each dataset, and the fixed values of $\beta = 0.01$ and $\alpha = 0.1$ were used for all datasets.

3.4.1 First Dispersion Measure:MSE

The results using the first dispersion measure, MSE, are reported in Table 3.1. The first column, MSE-only, indicates how the totally unsupervised approach of clustering based on only the unlabeled training data would perform. The second column, GINI-only shows how the completely supervised approach of clustering using the GINI index on the labeled training data performs. The third column is the proposed approach using both the MSE and GINI based on the labeled training data. The forth column indicates how MSE+GINI performs transductive inference when all the available data is used. A **bold** number is the minimum error for a given dataset, an *italic* number indicates that the result is significantly different from the transduction result. The totally unsupervised MSE-only approach always performs significantly worse than any of the supervised methods. Surprisingly, the GINI-only complete supervised approach was the best on four of the eight datasets. The transductive MSE+GINI method based on all available data showed no consistent improvements over the induction approach. This is consistent with other researchers who have reported that doing transductive inference using a regression estimate where the variance estimate was based on all the available data (both labeled and unlabeled) actually degraded results [20].

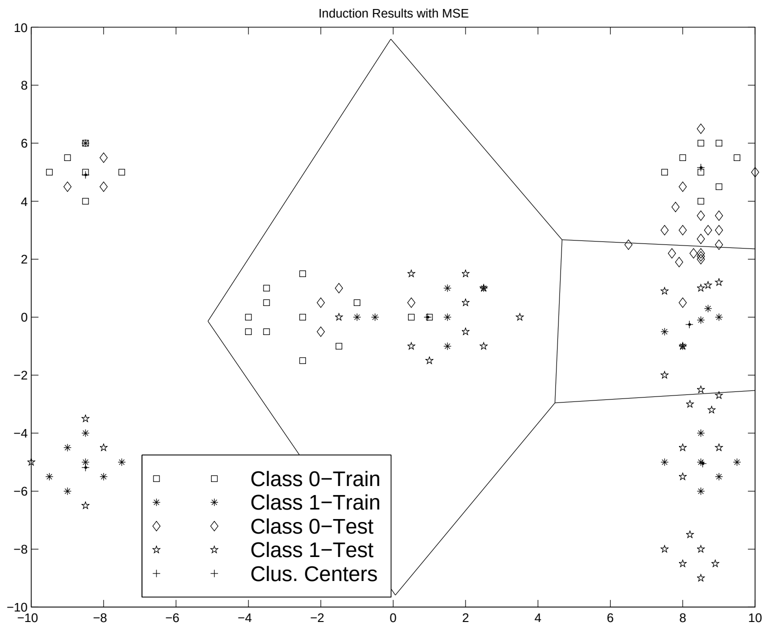
The MSE based approaches showed poor performance. To examine why consider the results of the MSE-based fitness functions on the cartoon example shown in Figure 3.1. The top plot shows the induction result (using just labeled data) and the bottom plot shows the transduction result (using both labeled and unlabeled data) cases. Note that on the center cluster transduction does work appropriately.

[2]The (k, β, α) values applied for each dataset were bright (15, 0.01,0.99), sonar (7,0.1,1), heart (7,0.25,0.75), ionosphere (7, 0.01,0.99), house (7,0.1,0.9), housing (11,0.01, 0.99), prognosis (11,0.4,0.6), and pima (11,0.01,0.99).

The inductive MSE+GINI method does not separate the cluster in the center of the figure, because such separation will result an impure cluster on the right. Using the additional unlabeled data, the transductive MSE+GINI reduces cluster dispersion by splitting the center cluster (bottom of Figure 3.1). On the other hand, both the transductive MSE+GINI method does not catch the downward shift of the top right cluster. Because the added unlabeled points are roughly equal distance from two top right cluster centers, adding unlabeled data has little effect despite the fact that a natural gap exits between the two clusters. The MSE minimizes only the compactness of the clusters. It is necessary to find clusters that are both compact and well separated. The DBI index is much more effective in this regard and the computational results are greatly improved when this cluster dispersion metric is applied.

3.4.2 Second Dispersion Measure:DBI

The DBI dispersion measure was much more effective than the MSE with regards to transduction. For the cartoon example, the top of the Figure 3.2 shows the resulting partition after using DBI in fitness function for the induction case. By using DBI, the method was able to catch the vertical shift within the data in transductive inference. The results for the DBI dispersion measure (Eq.3.7) on the UC-Irvine Data are reported in Table 3.2. The same experimental setup and parameters as mentioned above were used except that the maximum number of generations was set to 300. The same K was used as in MSE approach, but the regularization parameters were changed due to the different magnitude of the DBI. As a purely unsupervised approach DBI-only was even worse than MSE-only at classification. The inductive DBI+GINI approach was not significantly different from the GINI-Only approach. The transductive DBI+GINI was either better or not significantly worse than GINI-only approach. Transductive DBI+GINI consistently produced the best classification results of all the seven approaches tested primarily due to the more compact and better separated clusters found by DBI over MSE. The evidence indicates that capacity control based on both labeled and unlabeled data is much more effective using the DBI criterion than MSE.



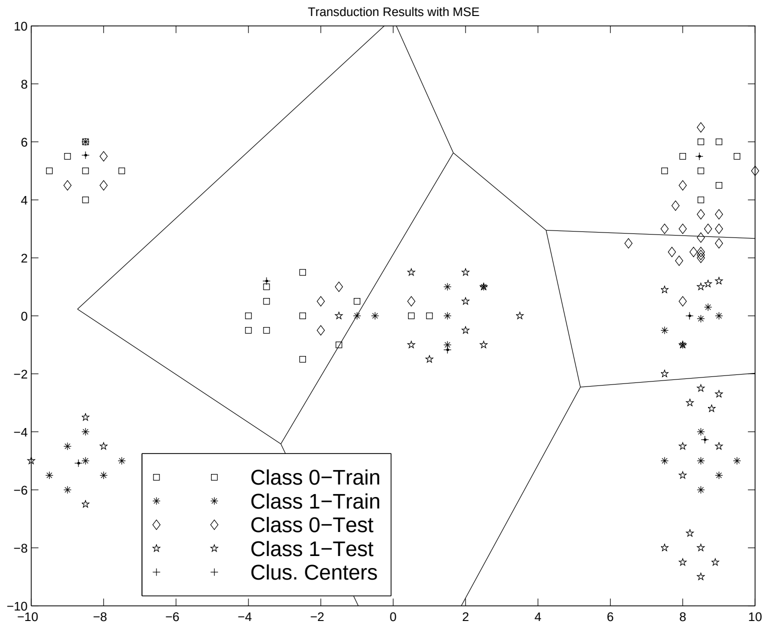
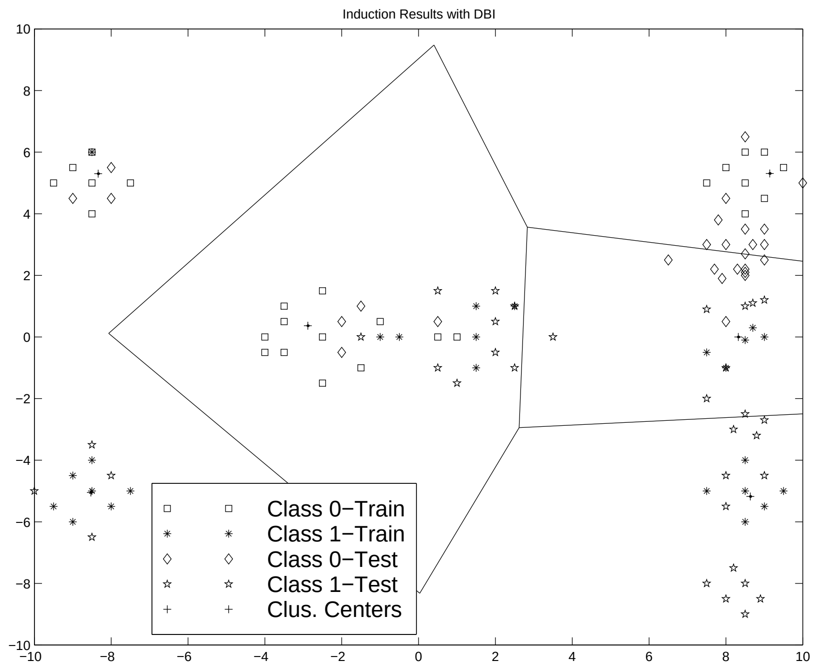


Figure 3.1: Induction and Transduction Results using MSE



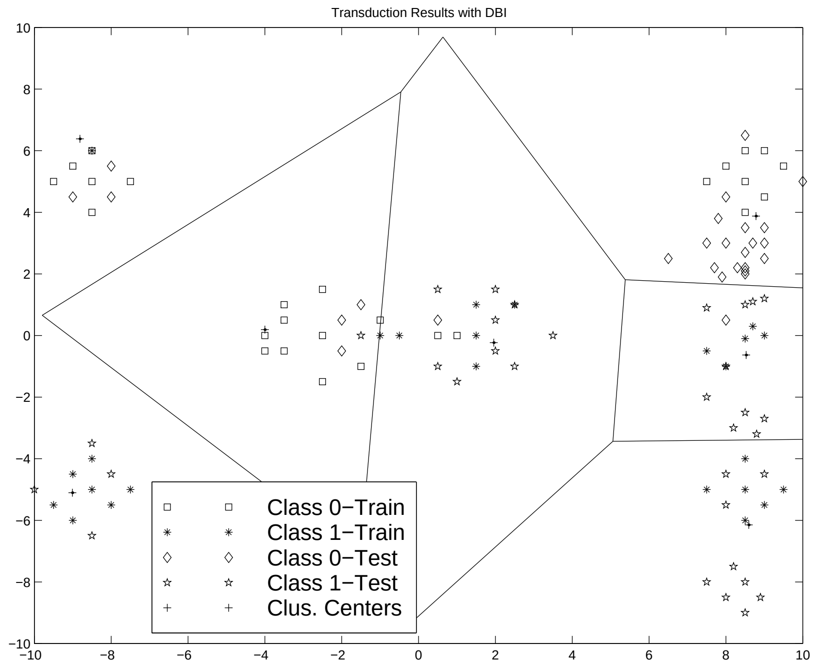


Figure 3.2: Induction and Transduction Results using DBI

Another finding from the results is that DBI dispersion measure favors a fewer number of non-empty clusters compared to the MSE dispersion measure. On average, using DBI dispersion measure resulted in 53.33%, 33.33%, 28.36%, 17.91%, 30.84%, 33.33%, 51.11% and 50.91% fewer non-empty clusters than MSE dispersion measure for transductive inference on the UCI datasets respectively.

In Table 3.3, results from some other classification techniques are compared with transductive DBI+GINI – specifically 3-nearest-neighbor classifier, linear and quadratic discriminant classifiers. The discriminant analysis was done using the SAS procedure DISCRIM [86]. All results are reported on the test datasets. The DBI-GINI is consistently one of the better methods.

3.5 Conclusion

A novel method for semi-supervised learning that combines aspects of supervised and unsupervised learning techniques has been introduced in this chapter. The basic idea is to take an unsupervised clustering method, label each cluster with the class membership, and simultaneously optimize the misclassification error of the resulting clusters. The intuition behind this approach is that the unsupervised component of the objective function acts as a form of regularization or capacity control during supervised learning to avoid overfitting. The objective function now is a linear combination of a measure of cluster dispersion and a measure of cluster impurity. The method can exploit any available unlabeled data during training since the cluster dispersion measure does not require class labels. This allows the approach to be used for transductive inference, the process of constructing a classifier using both the labeled training data and the unlabeled test data. Experimental results also show that using Davies-Bouldin Index for cluster dispersion instead of Mean Square Error improves transductive inference. This is due to the compact and well separated clusters found by minimizing DBI. DBI finds solution using much fewer clusters than MSE with much greater accuracy. The basic ideas in this chapter: incorporating classification information into an unsupervised algorithm and using the resulting algorithm for transductive inference are applicable to many types of unsupervised learning and are promising areas of future research.

Table 3.1: Results Using MSE in Fitness Function

	Induction			Transduction
Data Set	MSE-Only	GINI-Only	MSE+GINI	MSE+GINI
Bright	*0.06585*	***0.01084***	0.02507	0.02263
Sonar	*0.43279*	0.2541	***0.22951***	0.26066
Heart	*0.23636*	*0.21477*	0.2	**0.19659**
Iono.	*0.25673*	0.14423	**0.12788**	0.12981
Housing	*0.25828*	***0.15629***	*0.18874*	0.16887
House	*0.09846*	0.06692	**0.06**	0.06308
Prognos.	*0.1*	**0.05059**	*0.06235*	0.05235
Pima	*0.32402*	***0.27118***	0.30131	0.30393

	Induction			Transduction
Data Set	DBI-Only	GINI-Only	DBI+GINI	DBI+GINI
Bright	*0.26897*	**0.01084**	*0.01992*	0.01165
Sonar	*0.50656*	0.2541	*0.27049*	**0.23771**
Heart	*0.3841*	*0.21477*	*0.21136*	**0.19155**
Iono.	*0.34327*	0.14423	**0.12885**	0.13558
Housing	*0.4563*	0.15629	*0.17086*	**0.15497**
House	*0.11769*	**0.06692**	0.07462	0.06923
Prognos.	*0.38059*	*0.05059*	*0.04941*	**0.04353**
Pima	*0.34585*	**0.27118**	0.28428	0.28122

Table 3.2: Results Using DBI in Fitness Function

Data Set	3-NN	LinDisc	QuadDisc	DBI+GINI
Bright	*0.01247*	*0.02387*	*0.02112*	**0.01165**
Sonar	***0.2098***	*0.38025*	*0.35256*	0.23771
Heart	0.19773	***0.1745***	*0.22334*	0.19155
Iono.	*0.18846*	0.14624	**0.1294**	0.13558
Housing	*0.16291*	0.16013	*0.19946*	**0.15497**
House	0.06154	***0.0414***	0.06995	0.06923
Prognos.	**0.04235**	0.04797	*0.05348*	0.04353
Pima	0.28777	***0.2313***	*0.26401*	0.28122

Table 3.3: Comparison between Transductive DBI+GINI and 3-NN, LD, and QD

The supervised clustering method discussed in this chapter can also be generalized to regression problems. One could also implement scaling for variable selection. For each variable, a gene must be introduced to represent the scaling factor. Scaling can be either implemented within the same genome or using another genome for scaling factors within the GA.

In the next chapter we focus on developing capacity control techniques for supervised learning by incorporating unlabeled data into supervised learning task based on Support Vector Machines.

CHAPTER 4
Optimization Approaches to Semi-Supervised Learning

4.1 Introduction

The focus of this chapter is mathematical programming approaches to semi-supervised learning for classification tasks based on SVMs. The main idea of semi-supervised learning is to construct a classifier using both a training set of labeled data and a working set of unlabeled data. If none of the labels are known then the problem becomes clustering. If some of the labels are known, then the problem is classification. This chapter is based on the work reported in [9, 37].

There are many practical domains in which unlabeled data are abundant but labeled data are expensive to generate and therefore relatively scarce (e.g. medical diagnosis, web search, drug design, and database marketing). When the training data consist of relatively few labeled data points in a high-dimensional space, something must be done to prevent the classification or regression function from overfitting the training data. The key idea is that by exploiting the unlabeled data we hope to be able to provide additional information about the problem that can be used to improve accuracy on data with unknown labels (generalization) through capacity control with unlabeled data.

By including the unlabeled data in the testing set (working set), semi-supervised learning can be used to perform transductive learning instead of typical inductive learning. In induction, the task is to construct a good discriminant function valid everywhere. This function is fixed and applied to any future test data (Figure 2.1). In transduction, the labeled training data and unlabeled testing data are given, then the discriminant function is constructed based on all the available data. The learning task is to predict the labels of only those specific test data points, not all possible future points. This simpler task can result in theoretically better bounds on the generalization error [99], thus reducing the amount of required labeled data for good generalization (Figure 2.2).

For semi-supervised SVM we consider all possible labels of the test data and assign the labels that produce the best SVM with maximum margin based on all the available data, both labeled and unlabeled. For the purpose of this chapter we limit our discussion to linear SVM, but these methods can be extended to nonlinear support vector machines using the standard SVM approach of including kernel functions [99, 65]. In Section 4.2, we review SVMs to give foundation to develop semi-supervised methodology. In Section 4.3 we provide a general framework for viewing the semi-supervised support vector machine problem. Depending on how we penalize unlabeled data appearing in the margin the problem can be formulated as a linear or convex quadratic program with additional equilibrium constraints, mixed-integer constraints, or nonconvex objective terms. In Section 4.4 we examine practical approaches using the linear mixed integer program (MIP) formulation first introduced in [9]. By incorporating the MIP within a local learning framework, performance is greatly enhanced. In Section 4.5 we examine practical algorithms for a nonconvex quadratic formulation. Finally, we conclude this chapter with a brief summary and discussion of optimization issues in semi-supervised learning.

Other researchers have reported favorable results on semi-supervised methods on web-based text classification problems, for example using an EM (Expectation-Maximization) [74, 67], co-training in Bayesian networks [18], and a transductive version of SVM-Light [60]. Cataltepe and Magdon-Ismail [28] propose augmented error, which has components from both labeled and unlabeled data. They provide an analytical solution in the case of linear, noisy targets and linear hypothesis functions. They also show some results for the non-linear case. Theoretical results exist [27] on the relative value of labeled and unlabeled data.

4.2 Review of SVM Problem Formulation

The underlying problem of interest is to estimate a classification function $f : R^p \rightarrow \{\pm 1\}$ using input-output training data from two classes

$$(x_1, y_1), \ldots, (x_n, y_n) \in R^p \times \{\pm 1\}. \tag{4.1}$$

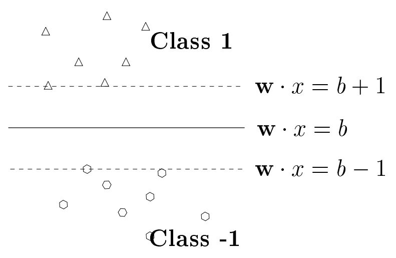


Figure 4.1: Optimal Plane Maximizes Margin

The function f should correctly or almost correctly classify unseen examples $(\mathbf{x}, y)$, i.e. $f(\mathbf{x}) = y$ if $(\mathbf{x}, y)$ is generated from the same underlying probability distribution as the training data. In this section we limit discussion to linear classification functions. If the points are linearly separable, then there exist an p-vector $\mathbf{w}$ and scalar b such that

$$\begin{array}{ll} \mathbf{w} \cdot x_i - b \geq 1 & if\ y_i = 1,\ and \\ \mathbf{w} \cdot x_i - b \leq -1 & if\ y_i = -1,\ i = 1, \ldots, n \end{array} \tag{4.2}$$

or equivalently

$$y_i[\mathbf{w} \cdot x_i - b] \geq 1,\ i = 1, \ldots, n. \tag{4.3}$$

The "optimal" separating plane, $\mathbf{w} \cdot x = b$, is the one that is furthest from the closest points in the two classes. Geometrically this is equivalent to maximizing the separation margin or distance between the two parallel planes $\mathbf{w} \cdot x = b + 1$ and $\mathbf{w} \cdot x = b - 1$ (see Figure 4.1).

The "margin of separation" in Euclidean distance is $2/\left\|\mathbf{w}\right\|_2$ where $\left\|\mathbf{w}\right\|_2 = \sqrt{\sum_{i=1}^{p} \mathbf{w}_i^2}$ is the 2-norm. To maximize the margin, we minimize $\left\|\mathbf{w}\right\|_2/2$ subject to the constraints (4.3). According to structural risk minimization, for a fixed empirical misclassification rate, larger margins should lead to better generalization and prevent overfitting in high-dimensional attribute spaces [98]. The classifier is called a support vector machine because the solution depends only on the points (called support vectors) located on the two supporting planes $\mathbf{w} \cdot x = b - 1$ and $\mathbf{w} \cdot x = b + 1$.

In general the classes will not be linearly separable, so the generalized optimal plane problem (4.4) [32, 98] is used. A slack term η_i is added for each point such that if the point is misclassified, $\eta_i \geq 1$. The quadratic programming formulation is (SVM-QP):

$$\begin{aligned} \min_{\mathbf{w},b,\eta} \quad & C\sum_{i=1}^{n}\eta_i + \frac{1}{2}\|\mathbf{w}\|^2 \\ s.t. \quad & y_i[\mathbf{w}\cdot x_i - b] + \eta_i \geq 1 \\ & \eta_i \geq 0, \quad i = 1,\ldots,n \end{aligned} \tag{4.4}$$

where $C > 0$ is a fixed penalty parameter. The capacity control provided by the margin maximization can greatly improve generalization [100, 97]. Typically, the following dual form of (4.4) is solved in practice:

$$\begin{aligned} \min_{\alpha} \quad & \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} y_i y_j \alpha_i \alpha_j x_i \cdot x_j - \sum_{i=1}^{n}\alpha_i \\ s.t. \quad & \sum_{i=1}^{n}\alpha_i y_i = 0 \\ & 0 \leq \alpha_i \leq C \quad i = 1,\ldots,n \end{aligned} \tag{4.5}$$

The Robust Linear Programming approach to SVM is identical to SVM-QP except the margin term is changed from the 2-norm $\|\mathbf{w}\|_2$ to the 1-norm, $\|\mathbf{w}\|_1 = \sum_{j=1}^{p}|w_j|$. The problem becomes the following robust linear program (SVM-RLP) [11, 21, 8]:

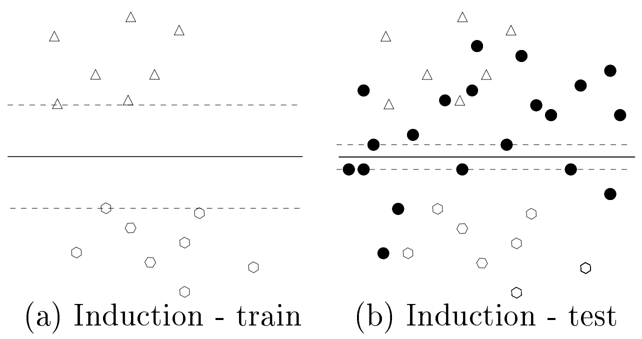
$$\begin{aligned} \min_{\mathbf{w},b,s,\eta} \quad & C\sum_{i=1}^{n}\eta_i + \sum_{j=1}^{p}s_j \\ s.t. \quad & y_i[\mathbf{w}\cdot x_i - b] + \eta_i \geq 1 \\ & \eta_i \geq 0, \quad i = 1,\ldots,n \\ & -s_j <= \mathbf{w}_j <= s_j, \quad j = 1,\ldots,p. \end{aligned} \tag{4.6}$$

The RLP formulation is a useful variation of SVM with some nice characteristics. The 1-norm weight reduction still provides capacity control. The results in [64] can be used to show that minimizing $\|\mathbf{w}\|_1$ corresponds to maximizing the separation margin using the infinity norm. Statistical learning theory can be extended to incorporate alternative norms. For example, an adaptive weighted euclidian norm is implemented in the structural risk minimization framework [94]. One major benefit of SVM-RLP over SVM-QP is dimensionality reduction. Both SVM-RLP and SVM-QP minimize the magnitude of the weights $\mathbf{w}$. But SVM-RLP forces more of the weights to be 0 due to the properties of the 1-norm. This results in dimensionality reduction since variables with 0 weights can be removed from the model. Another benefit of SVM-RLP over SVM-QP is that it can be solved using linear programming instead of quadratic programming.

SVMs are easily generalized to nonlinear discriminants through the introduction of kernel functions [99, 65]. The basic idea is that the data are mapped nonlinearly to a higher dimensional space and a linear SVM is constructed in the transformed space corresponding to a nonlinear classifier in the original space. We limit our formulation to the linear classification problem in this section and leave computational studies of these approaches extended with kernels to later sections.

4.3 Semi-supervised SVM

Our semi-supervised support vector machine approach can be illustrated by a simple example. Consider the two-class problem shown in Figure 4.2(a). Since the labeled training sets are linearly separable, there exists an infinite number of possible separating planes that correctly classify the two sets. Intuitively, the best linear classifier is the middle plane shown that separates the two sets with the greatest margin. The margin is the sum of distances from the closest points (the support vectors) in each set to the plane or equivalently the distance between the supporting planes for each set. The supporting planes are shown using dotted lines. Statistical Learning Theory proves that for a given misclassification error, maximizing the margin of separation minimizes a bound on the expected misclassification error on future unseen data [99]. Maximizing the margin reduces the capacity of the function to fit data. Intuitively, a "fat" plane with wide margin has less capacity to fit data than a "skinny" one. In SVM, the optimal plane can be found using


(a) Induction - train (b) Induction - test

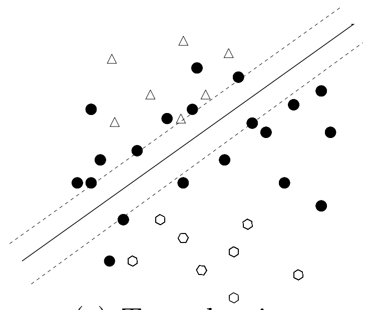

(c) Transduction

Figure 4.2: Traditional SVM (a, b) versus Semi-Supervised SVM (c)

quadratic or linear programming depending on the metric used to measure the margin distance [99, 98, 4]. Consider now the additional unlabeled test data shown in Figure 4.2(b). The SVM performs poorly on this particular test set in terms of classification accuracy of the testing data. Note also that the resulting margin for the combined labeled training data and unlabeled testing data is very small. If we construct the SVM margin that correctly classifies the training data and achieves the widest margin based on all the data, the results found by our semi-supervised SVM are significantly improved and the preferable plane is shown in Figure 4.2(c). Results in statistical learning theory show that, for a fixed misclassification error, maximizing the margin based on all the data (train and test) can lead to better bounds on the expected generalization error [99].

The basic idea of semi-supervised support vector machines is that we want the best support-vector machine on the labeled data that has no or very few unlabeled points in the margin. Thus we want to penalize the support vector machine if unlabeled points fall in the margin. Specifically, we define the semi-supervised support vector machine problem (S^{3}VM) as:

$$\begin{array}{ll}\min\limits_{\mathbf{w},b,\eta,\xi,z} & C\left[\sum\limits_{i=1}^{n}\eta_i+\sum\limits_{j=n+1}^{n+\ell}g(\mathbf{w}\cdot x_j-b)\right]+\parallel\mathbf{w}\parallel \\ s.t. & y_i(\mathbf{w}\cdot x_i-b)+\eta_i\geq 1 \quad \eta_i\geq 0 \quad i=1,\dots,n\end{array} \tag{4.7}$$

where $C > 0$ is a fixed misclassification penalty parameter and $g(\alpha)$ is the margin penalty function on unlabeled data x_j $j = n+1, \dots, n+\ell$.

The question then is how to define g. For a hard margin approach in which no unlabeled points are allowed in the margin, the margin penalty function is defined as

$$\begin{array}{rll} g_\infty(\alpha) := & \infty & for \ -1<\alpha<1 \\ & 0 & otherwise\end{array} \tag{4.8}$$

If an unlabeled point falls outside the margin, it is considered well-classified and no penalty is incurred.

We can transform the hard margin g_∞ problem into a linear or quadratic program with an additional equilibrium constraint. We start with either SVM formulation, (4.4) or (4.6), and then add two constraints for each point in the working set. One constraint calculates the misclassification error as if the point were in class 1 and the other constraint calculates the misclassification error as if the point were in class -1. We add a constraint that forces one of the two misclassification errors per point to be zero. This produces the following mathematical programming problem with equilibrium constraints:

$$\begin{aligned}\min_{\mathbf{w},b,\eta,\xi,z} \quad & C\left[\sum_{i=1}^{n}\eta_i\right] + \parallel \mathbf{w} \parallel \\ s.t. \quad & y_i(\mathbf{w}\cdot x_i - b) + \eta_i \geq 1 \quad \eta_i \geq 0 \quad i = 1,\dots,n \\ & \mathbf{w}\cdot x_j - b + \xi_j \geq 1 \quad \xi_j \geq 0 \quad j = n+1,\dots,n+\ell \\ & -(\mathbf{w}\cdot x_j - b) + z_j \geq 1 \quad z_j \geq 0 \\ & \xi_j \cdot z_j = 0 \quad j = n+1,\dots,n+\ell\end{aligned} \tag{4.9}$$

The requirement that no unlabeled points may fall in the margin may be too strong. A natural relaxation of the problem would be to move the equilibrium constraint into the objective and use it as the margin penalty function g. This results in the following nonconvex quadratic optimization problem:

$$\begin{aligned}\min_{\mathbf{w},b,\eta,\xi,z} \quad & C\left[\sum_{i=1}^{n}\eta_i + \sum_{j=n+1}^{n+\ell}\xi_j \cdot z_j\right] + \parallel \mathbf{w} \parallel \\ s.t. \quad & y_i(\mathbf{w}\cdot x_i + b) + \eta_i \geq 1 \quad \eta_i \geq 0 \quad i = 1,\dots,n \\ & \mathbf{w}\cdot x_j - b + \xi_j \geq 1 \quad \xi_j \geq 0 \quad j = n+1,\dots,n+\ell \\ & -(\mathbf{w}\cdot x_j - b) + z_j \geq 1 \quad z_j \geq 0\end{aligned} \tag{4.10}$$

Close examination of this choice of error function shows that it has attractive properties. If the unlabeled point x_j falls outside or on the margin then ξ_j or z_j is 0, and there is no error associated with that point. If the point falls in the margin, then for $\gamma = w \cdot x_j - b$, $\xi_j = 1 - \gamma$ and $z_j = 1 + \gamma$ by construction of the support vector machine. The following piecewise quadratic margin penalty function is produced (see Figure 4.3(a)):

$$\begin{aligned} g_2(\gamma) := \; & 1 - \gamma^2 \quad for \;\; -1 < \gamma < 1 \\ & 0 \qquad\quad otherwise. \end{aligned} \tag{4.11}$$

Another natural choice would be a margin penalty function that calculates the minimum of the two possible misclassification errors. The final class of a point corresponds to the one that results in the smallest error. This is the transductive

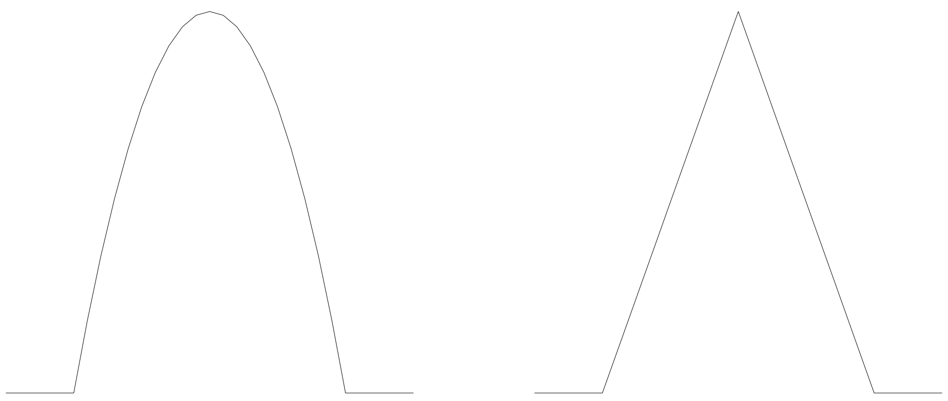

(a) $g_2(\gamma)$ (b) $g_1(\gamma)$

Figure 4.3: Margin Penalty Functions

idea as proposed by Vapnik [99]. It has the advantage that if the correct labels are found, the resulting SVM will be identical to the one produced if the points were known. The minimum error formulation is [9]:

$$
\begin{array}{ll}
\min\limits_{\mathbf{w},b,\eta,\xi,z} & C\left[\sum\limits_{i=1}^{n}\eta_i + \sum\limits_{j=n+1}^{n+\ell}\min(\xi_j, z_j)\right] + \parallel \mathbf{w} \parallel \\
s.t. & y_i(\mathbf{w}\cdot x_i + b) + \eta_i \geq 1 \quad \eta_i \geq 0 \quad i = 1,\ldots,n \\
 & \mathbf{w}\cdot x_j - b + \xi_j \geq 1 \quad \xi_j \geq 0 \quad j = n+1,\ldots,n+\ell \\
 & -(\mathbf{w}\cdot x_j - b) + z_j \geq 1 \quad z_j \geq 0
\end{array}
\tag{4.12}
$$

The resulting margin penalty function is shown in Figure 4.3(b)

$$
\begin{array}{lll}
g(\gamma) = g_1(\gamma) := & 1 - |\gamma| & for \ -1 < \gamma < 1 \\
 & 0 & otherwise
\end{array}
\tag{4.13}
$$

For our experimental study of practical methods for solving these problems we focused on the minimum error formulation (4.12). But this is not to say that other formulations are not possible or preferable. In the next two sections we explore two different approaches to practically solving this problem.

4.4 Mixed-Integer Programming Formulation

Integer programming can be used to exactly solve S^{3}VM (4.12). The basic idea is to add a 0 or 1 decision variable, d_j, for each point $\mathbf{x}_j$ in the working set. This variable indicates the class of the point. If $d_j = 1$ then the point is in class 1 and if $d_j = 0$ then the point is in class -1. This results in the following mixed integer program (S^{3}VM-MIP):

$$\begin{array}{lrll}
\min\limits_{\mathbf{w},b,\eta,\xi,z,d} & C\left[\sum\limits_{i=1}^{n}\eta_i + \sum\limits_{j=n+1}^{n+\ell}(\xi_j + z_j)\right] + \parallel \mathbf{w} \parallel_1 & & \\
s.t. & y_i(\mathbf{w}\cdot x_i - b) + \eta_i \geq 1 & \eta_i \geq 0 & i = 1,\dots,n \\
& \mathbf{w}\cdot x_j - b + \xi_j + M(1-d_j) \geq 1 & \xi_j \geq 0 & j = n+1,\dots,n+\ell \\
& -(\mathbf{w}\cdot x_j - b) + z_j + Md_j \geq 1 & z_j \geq 0 & d_j = \{0,1\}
\end{array} \tag{4.14}$$

The constant $M > 0$ is chosen sufficiently large such that if $d_j = 0$ then $\xi_j = 0$ is feasible for any optimal $\mathbf{w}$ and b. Likewise if $d_j = 1$ then $z_j = 0$. In this chapter, we use the 1-norm of $\mathbf{w}$ in the objective.

A globally optimal solution to this problem can be found using CPLEX or other commercial mixed integer programming codes [33] provided computer resources are sufficient for the problem size. Using the mathematical programming modeling language AMPL [47], we were able to model S^{3}VM easily and solve it using CPLEX. One practical limitation of this approach is the capacity of the MIP solver used. Using CPLEX 4.0 on a Sun Ultra 1 with 700MB RAM we found it was practical to include at most 50 unlabeled data points due to the CPU time limitation.

4.4.1 Local Semi-Supervised Support Vector Machines

To get around the practical restriction on the number of integer variables and thus unlabeled data can be handled by our MIP solver, we utilized the S^{3}VM-MIP as part of a local learning algorithm. In local learning, a point is classified based on points in its "neighborhood". For example, in the K-Nearest-Neighbor algorithm (K-NN), the K nearest neighbors to a point (by Euclidean distance or some other metric) are found and then the point is assigned the majority class of the K nearest neighbors. Local learning methods are often called memory-based methods, because training examples are kept in “memory” and used to classify new points. Since the local models have fewer training examples, it takes much less computational time to optimize each local S^{3}VM than to train one global one at the expense of many local models. Previous empirical studies have shown that the generalization ability of local methods often exceeds that of global ones since the local models include only the points which are related to the query point (interested unlabeled data) in a given learning task. Many variations exist for both selecting the neighborhoods and determining the output class based on the neighbor. For example, Discriminant Adaptive Nearest Neighbor [54] uses local discriminant analysis to estimate the class within K-NN classification. Lawrence et al. [63] use local neural network models for function approximation. See [1] for a survey of approaches.

4.4.2 Local S^{3}VM and Experimental Results

Local S^{3}VM is nothing but an application of S^{3}VM in a local neighborhood of each unlabeled point as determined by the K-NN algorithm using Euclidean distance. This neighborhood includes both labeled and unlabeled examples. In order to have enough labeled examples in each neighborhood, we arbitrarily pick K as 10% of all available data points. Further study is needed on how to best select the neighborhood of a point. We can summarize the method (Local-S^{3}VM) to classify a given unlabeled point in the following steps:

1. Find K-NN for a given unlabeled point.
2. If all the labeled points in the neighborhood are in one class, then label the unlabeled point as in that class and end. Otherwise continue.
3. Solve the S^{3}VM-MIP (4.14) in the neighborhood.
4. Label the point according to the result of S^{3}VM .

There are many advantages to using Local S^{3}VM over using a single global S^{3}VM. In transduction for any data, we need to construct a new model. So the fact that local S^{3}VM must compute a new model for each point is also true for any transductive algorithm. Although there are as many models as unlabeled points

Table 4.1: Dataset Summary Statistics

Data Set	Dim	Points	Test-size
Bright	14	2462	50*
Cancer	9	699	70
Diagnostic	30	569	57
Dim	14	4192	50*
Heart	13	297	30
Housing	13	506	51
Ionosphere	34	351	35
Musk	166	476	48
Sonar	60	208	21
Pima	8	769	50*

to solve in Local S^3VM , the overall computational time of the algorithm including time to find the local neighborhood is generally less than the global S^3VM algorithm. This is because we have fewer unlabeled points in each local model which means we have fewer binary variables in each model. Having fewer binary variables results in less running time for each local model. Another advantage is that the overall classification function by Local S^3VM is nonlinear (piecewise linear to be exact) when a linear S^3VM is used locally.

Determining nearest neighbors of a point can become problematic for large datasets. One must consider an appropriate metric and method to find K-NN. Since we use datasets which have relatively small dimensions, we use Euclidean distance combined with a partial sort algorithm [71] to find the local neighborhood. As mentioned in the outlines of the algorithm, for each unlabeled point, a related data file is created and the S^3VM model is solved using AMPL. Then the output of AMPL is analyzed to find the label of the point.

Our computational study of S^3VM consisted of 10 trials using the ten real-world data sets described in Table 4.1 (eight from [68] and the bright and dim galaxy sets from [75]) [3]. The basic properties of the datasets are summarized in Table 4.1. Each dataset is sampled randomly 10 times and each working set (test set) is composed of 10% of the data except the Bright, Dim, and Pima datasets in

[3]The continuous response variable in Housing dataset was categorized at 21.5

Table 4.2: Average Error Results for Inductive and Transductive SVM Methods

Data Set	SVM-RLP	S^3VM	Local SVM	Local S^3VM	3-NN
Bright	0.02	0.018	0.008	$\underline{0.006}$	0.028
Cancer	0.036	$\underline{0.034}$	0.06	0.059	$\underline{0.034}$
Diagnostic	0.035	$\underline{0.033}$	0.039	0.039	0.039
Dim	0.064	0.054	$\underline{0.042}$	0.044	0.074
Heart	0.173	$\underline{0.16}$	0.257	0.253	0.17
Housing	0.155	0.151	$\underline{0.118}$	0.124	0.177
Ionosphere	0.109	$\underline{0.106}$	0.117	0.109	0.129
Musk	0.173	0.173	0.092	$\underline{0.085}$	0.208
Sonar	0.281	0.219	0.181	$\underline{0.143}$	0.171
Pima	0.22	0.222	0.22	$\underline{0.218}$	0.264

which the size of the working set is set to 50 points and rest of the data are used as the training set. We use the following formula to pick the penalty parameter: $C = \frac{(1-\lambda)}{\lambda(n+\ell)}$ with $\lambda = 0.001$, n is the size of the training set, and ℓ is the size of the working set. The average working set errors are reported in Table 4.2. The best result from different models is underlined for each dataset.

Columns two and three of Table 4.2 provide a comparison of the inductive linear 1-norm support vector machine (SVM-RLP 4.6) with the transductive linear 1-norm SVM optimized used mixed integer programming (S^{3}VM-MIP 4.14). On all ten datasets, the transductive S^{3}VM-MIP results are either slightly better or not significantly different than the inductive results found using SVM-RLP. Note that all parameters of the formulations are identical; the only difference between the two formulations is the use of unlabeled data for the transductive case. For this formulation, unlabeled data seems to help and never hurt generalization.

Columns 4 and 5 of Table 4.2 compare an inductive version of Local SVM and the transductive version of Local S^{3}VM . In our study, the neighborhoods of points used by both Local SVM and Local S^{3}VM are identical. Thus for each testing set point the optimization problem solved by local S^{3}VM is identical to the one solved by local SVM once the terms involving the unlabeled data are removed. This was done to ensure that the introduction of unlabeled data was the only change in the experiment. But in fact, it means the unlabeled data are being used to determine the effective size of the neighborhood for Local SVM which in itself is a form of transduction. Column 6 of Table 4.2 gives results for the 3-nearest-neighbor algorithm. This was done to examine improvements that occur by simply switching to a local algorithm. Local S^{3}VM outperformed or did as well as Local SVM on eight of the ten datasets, once again supporting the transductive hypothesis. The improvements cannot be simply attributed to a local learning strategy since 3-NN did worse than both Local SVM and S^{3}VM on nine of ten datasets.

Overall, Local S^{3}VM was consistently the best or almost the best in our experiments. Either S^{3}VM or the Local S^{3}VM obtained the best results on most of the datasets except Dim and Housing datasets. The results indicate that using the labeled and unlabeled points in a transduction model can improve accuracy. Local S^{3}VM resulted in better accuracy than S^{3}VM on six datasets. One noteworthy point is that in some cases (Sonar, Musk, Housing, Bright) Local S^{3}VM improved accuracy notably. On Cancer, Diagnostic, Heart, and Ionosphere the fact that S^{3}VM performed best indicates that if the neighborhood of Local S^{3}VM is increased, Local S^{3}VM could perform better. The best method of choosing neighborhoods for local methods is still very much an open question. The proposed algorithm in this section takes into consideration only one unlabeled point at a given time. Although there might be many unlabeled points in a given neighborhood, the algorithm returns the results only on the test point of interest. The results for other points are basically discarded. One extension would be keeping these results for a final vote at the end of the algorithm. In this case, we can assign a probability of class membership for a certain point. The results from one point can also be used as starting points to improve the solution time of Local S^{3}VM on nearby points.

4.5 Nonconvex Quadratic Approach

An alternative approach to solving the minimum error S^{3}VM problem (4.12) is to convert it into a nonconvex quadratic program. We adapt the approach used previously to handle disjunctiveness of classification labels within the bilinear separability [12] and global tree optimization problems [17, 3, 12]. Once again a decision variable d_j is introduced for each point such that at optimality if $d_j = 1$ then the predicted class of x_j is 1 and if $d_j = 0$ then the x_j is predicted as class -1. The resulting problem is (S^{3}VM-QP)

$$\begin{array}{lll} \min\limits_{\mathbf{w},b,\eta,\xi,z,d} & C\left[\sum\limits_{i=1}^{n}\eta_i + \sum\limits_{j=n+1}^{n+\ell}(d_j\xi_j + (1-d_j)z_j)\right] + \frac{1}{2} \parallel \mathbf{w} \parallel^2 & \\ s.t. & y_i(\mathbf{w}\cdot x_i - b) + \eta_i \geq 1 \quad \eta_i \geq 0 \quad i = 1,\dots,n & (4.15) \\ & \quad \mathbf{w}\cdot x_j - b + \xi_j \geq 1 \quad \xi_j \geq 0 \quad j = n+1,\dots,n+\ell & \\ & -(\mathbf{w}\cdot x_j - b) + z_j \geq 1 \quad z_j \geq 0 \quad 0 \leq d_j \leq 1 & \end{array}$$

An intuitively simple approach is to adapt a block coordinate descent algorithm (e.g. [14]) which alternates between fixing d and estimating the SVM weights $\mathbf{w}, b$ and other dependent variables, and optimizing d with the other SVM variables fixed. In [17], it was shown for a class of problems that includes S^{3}VM-QP (4.15), using 2-norm $\parallel \mathbf{w} \parallel^2$ such an approach will converge in a finite number of iterations to a solution satisfying the minimum principal necessary optimality conditions. No linesearch is required. The proof in [17] does require each subproblem be solved to optimality, but this condition can be relaxed to require only a strict decrease in the objective function. On the global tree optimization problem [3, 17] , the block coordinate descent algorithm was found to be very prone to local minima so a tabu search method was used. When applied to transduction, we also found this simple algorithm to be very prone to local minima and thus do not report the results here. To improve the results, we developed a heuristic variation of the block coordinate descent algorithm. We introduce this algorithm in the following section.

4.5.1 A Descent Algorithm for Transductive SVM

The essential idea behind our heuristic approach is that we start by heavily penalizing solutions with points falling within the margin and then relax this requirement in order to find solutions with wider margin. Just as in the basic block coordinate descent method, we first estimate the labels ($d_j,\ j = n+1,\dots,n+\ell$) based on our current estimate of the SVM, and then solve S^{3}VM-QP with d fixed.

Note that in practice and for easy introduction of nonlinearity via kernels we solve the dual of Problem (4.15) which for fixed d reduces to the usual dual SVM problem (Eq. 4.5) tailored for transduction :

$$\begin{array}{ll} \min\limits_{\alpha} & \frac{1}{2}\sum\limits_{i=1}^{n+\ell}\sum\limits_{j=1}^{n+\ell} y_i y_j \alpha_i \alpha_j K(x_i, x_j) - \sum\limits_{i=1}^{n+\ell} \alpha_i \\ s.t. & \sum\limits_{i=1}^{n+\ell} \alpha_i y_i = 0 \\ & 0 \leq \alpha_i \leq C \quad i = 1, \ldots, n+\ell \end{array} \tag{4.16}$$

where $y_j = 2 * (d_j - \frac{1}{2})$ for $j = n+1, \ldots, n+\ell$, $K(\cdot,\cdot)$ is a kernel function. This process is repeated until a local minimum is reached. Then the weight on the misclassification error C is decreased allowing wider margins. In order to escape from local minima, the algorithm switches the labels of unlabeled data close to the separating hyperplane, if necessary. For this purpose, we check the consecutive solutions to track local minima. If 10 consecutive solutions are the same we assign the opposite labels to the points satisfying $|\mathbf{w} \cdot x_{i+1} - b| < S$. Occasionally a local minima is found with all points classified in one class ($\mathbf{w} = 0$). In this case, the algorithm restarts using the same initial conditions except for a reduced margin penalty parameter C for the unlabeled data. We empirically picked $C = \frac{\lambda}{100*(1-\lambda)}$, because it performed well in most cases. To ensure a good starting solution, the initial label assignments are made based on the closest class center for each unlabeled point. The resulting algorithm can be summarized as follows:

Algorithm 4.5.1. *S^3VM-IQP*

- *Find class centers from training points*
- *Assign labels d_0 to working set according to the closest class center*
- *Initialization: $i = 0$, $\lambda = 0.9$, $C = \frac{\lambda}{100*(1-\lambda)}$, $counter = 0$, $S = 0.2$.*
- *While $i \leq max\_iteration$*

```
1. Fix $d_i$ and solve Problem 4.15 (or its dual (4.16)) to find ($\mathbf{w}_{i+1}, b_{i+1}, \eta_{i+1},$
   $\xi_{i+1}, z_{i+1}$).
2. Fix ($\mathbf{w}_{i+1}, b_{i+1}, \eta_{i+1}, \xi_{i+1}, z_{i+1}$) and solve Problem 4.15 for $d_{i+1}$.
3. Check convergence criteria
     – If solution is same as the previous one
       then counter=counter +1 and $\lambda = \lambda * 0.9$
       else if there exists no point within margin
       then stop
       else let counter = 0
     – if counter > 10 then let counter = 0 and assign the opposite labels
       to the points satisfying $|\mathbf{w}_{i+1} \cdot x - b_{i+1}| < S$
     – if solution is all-in-one-class then reassign initial conditions
       except $i$ and let $\lambda * = 0.9$
4. $i = i + 1$
```

As a benchmark for transductive SVM, we report results from SVM-Light proposed by Joachims in [60, 59]. Transductive SVM-Light also can be viewed as a block coordinate descent algorithm that alternates between estimating the class labels and optimizing the SVM based on those labels. Transductive SVM-Light has an inner and an outer loop. The outer loop adjusts the penalty parameters on misclassification errors. Different errors are used for the unlabeled data according to their estimated class labels. After initial inductive iteration, the algorithm starts with low penalty terms for unlabeled data. Two penalty terms (C^*_-, C^*_+) are used in transductive SVM-Light, each for classifying an unlabeled point as a class -1 or a class 1 object respectively. Then it uniformly increases the influence of unlabeled data up to a user-defined penalty level. During this phase, the algorithm tunes these penalty terms in a way to satisfy a user-defined bias in data. The inner loop optimizes the SVM for the given penalties. The inner loop switches the labels of two given points, if such an action reduces the overall error. Like S^{3}VM-IQP, SVM-Light alternates the labels to avoid local minima. The primary difference is that SVM-Light changes the signs of at most two points at a time. Another

Table 4.3: Average Error Results for Transductive and Inductive Methods

Data Set	SVM-QP	SVM-Light	S^{3}VM-IQP
Heart	<u>0.16</u>	0.163	0.1966
Housing	0.1804	<u>0.1608</u>	0.1647
Ionosphere	<u>0.0857</u>	0.1572	0.0943
Sonar	0.1762	0.2524	<u>0.1572</u>

difference is SVM-Light uses different margin penalty parameters for class 1 and class -1 objects. In addition, unlike S^{3}VM-QP, it starts with lower values for margin penalty parameters. Details of SVM-Light and successful results on large datasets can be found in [60]. We use the default parameter options in our experiments with SVM-Light.

4.5.2 S^{3}VM-IQP Results

In this section we compare S^{3}VM-IQP with SVM-QP (Eq. 4.5) and transductive SVM-Light. We use the same datasets as in the previous section. Due to the long computational times for S^{3}VM-IQP and transductive SVM-Light, we limit our experiments to only the Heart, Housing, Ionosphere, and Sonar datasets. Linear kernel functions are used for all methods used in this section. The results given in Table 4.3 show that using unlabeled data in the case of datasets Heart and Ionosphere affects generalization ability slightly but the difference between the best transductive result and SVM-QP (Eq. 4.5) is not statistically significant. In the other two cases (Housing and Sonar), the best transductive method outperforms SVM-QP significantly. On two datasets S^{3}VM-IQP performs significantly better than transductive SVM-Light and in one case (Housing) the difference between two methods is not statistically significant.

As indicated above, the results from both S^{3}VM-IQP and SVM-Light are inconclusive. Both algorithms are much more expensive than their inductive versions. From the results on the mixed integer programming approaches we know that transduction can improve learning. We speculate that the reason that these improvements were not found using S^{3}VM-IQP and SVM-Light is that the optimization problem is very difficult and that the methods are failing to find the global minima. We know from the prior experiments that there is very little room for improvement on these specific learning tasks. Very few local minima will lead to better generalization. S^3VM-MIP and its local version are finding globally optimal solutions that are better. From the results on SVM-Light reported in [60] we know that on larger problems in text categorization, transductive inference using SVM-Light did lead to significant improvements. So on different learning tasks S^3VM-IQP may perform better as well. We speculate that on problems where there are many local minima that improve generalization, it is not as essential that the global minimum be found. Further studies are needed to identify when methods that find good but not globally optimial solutions are sufficient. Note that nonlinear kernels also might result in better generalization.

4.6 Conclusion

We examined mathematical models for semi-supervised support vector machines (S^3VM). We proposed a general S^3VM model that minimizes both the misclassification error and the function capacity based on all the available data. Three different functions for penalizing unlabeled points falling in the margin were discussed. Our computational investigation focused on the minimum error formulation for the transductive inference problem. We converted this problem to a mixed-integer program that can be exactly solved using commercial integer programming packages. By using the MIP formulation with a local learning algorithm, a powerful scalable transductive inference method was created. Our computational experiments found that the local learning method was the most effective overall. Further studies are needed to determine how to best select neighborhoods and to choose the parameters within the local S^3VM-MIP. In addition, very efficient computational methods for the local S^3VM-MIP are needed. One possibility is to use the estimated labels and models for one point as a starting point for other points. We also examined a noncovex quadratic optimization approach to S^3VM. Our computational studies were less conclusive using this approach. The best optimization approach for solving this problem is still very much an open question.

Increasingly competitive markets, challenging scientific problems, and complex decision processes require to use all available information on hand to enhance the output of current models. In the last two chapters we proposed methods to introduce new approaches for capacity control by solving semi-supervised learning problems. These methods were implemented in the feature space. We will shift our focus in the next chapter to propose capacity control techniques in the label space.

We proposed using unlabeled data as an extra information for current machine learning methods. One can also use the output of several models to improve the quality of the final output. Boosting [87] is a method for combining the output of the several machine learning models. In the next chapter, we use column generation technique from mathematical programming to boost decision stumps and decision trees.

CHAPTER 5
A Column Generation Algorithm for Boosting

5.1 Introduction

Recent papers [87] have shown that boosting, arcing, and related ensemble methods (hereafter summarized as boosting) can be viewed as margin maximization in label space. By changing the cost function, different boosting methods such as AdaBoost can be viewed as gradient descent to minimize this cost function [66]. Some authors have noted the possibility of choosing cost functions that can be formulated as linear programs (LP) but then dismiss the approach as intractable using standard LP algorithms [51, 81, 23]. In this chapter, we show that LP boosting is computationally feasible using a classic column generation simplex algorithm [72]. This method performs tractable boosting using any cost function expressible as an LP.

We specifically examine the variations of the 1-norm soft margin cost function used for support vector machines [82, 4, 65] (See SVM-RLP problem 4.6 in Chapter 4). One advantage of these approaches is that immediately the method of analysis for support vector machine problems becomes applicable to the boosting problem. In Section 5.2, we summarize boosting briefly and explain the motivation behind this study. We also review theoretical findings indicated in [10, 40]. In Section 5.3, we discuss the soft margin LP formulation adapted to boosting. By adopting linear programming, we immediately have the tools of mathematical programming at our disposal. By use of duality theory and optimality conditions, we can gain insight into how LP boosting works mathematically. In Section 5.4, we examine how column generation approaches for solving large scale LPs can be adapted to boosting.

For classification, we examine both standard and confidence-rated boosting. Standard boosting algorithms use weak learners (base learners, hypotheses) that are classifiers (such as decision trees, neural networks etc.), that is, whose outputs are in the set $\{-1, +1\}$. Schapire and Singer [88] have considered boosting weak learners whose outputs reflected not only a classification but also an associated confidence encoded by a value in the range $[-1, +1]$. They demonstrate that so-called confidence-rated boosting can speed convergence of the composite classifier, though the accuracy in the long term was not found to be significantly affected. In Section 5.5, we discuss the minor modifications needed for LPBoost to perform confidence-rated boosting.

The methods we develop can be readily extended to any boosting problem formulated as an LP. We demonstrate this by adapting the approach to regression in Section 5.6. Computational results and practical issues for implementation of the method are reported in Section 5.7.

5.2 Motivation for Soft Margin Boosting

As social beings, humans make decisions by asking friends their opinions. In some cases, we might simply make a decision/judgement based on the majority opinion. The same phenomenon is valid for democratic, civilized societies in general. Public decisions are made by the majority vote. As ordinary citizens, our rights to vote do not require sophisticated knowledge, advanced education *etc.*. Conventional wisdom suggests that even though we are not in the position to make public decisions as individuals, since it is representative of the concerned citizens, the majority vote will be the right choice. Surprisingly, voting methods also perform well in supervised learning cases. In voting methods, the idea is to construct several learning models and classify objects based on the majority vote of the outputs from these models.

There have been many successful variations of voting methods. The most famous one, the AdaBoost algorithm [87], is given below. r is the maximum number of boosting rounds given to the algorithm. a_j $j = 1, \ldots, r$ is the weight of the each weak learner (classifier), h_j, from the the class of functions, H. ϵ_j is the weighted error rate in each boosting iteration.

Algorithm 5.2.1 (AdaBoost).

```
Given as input training set: $S$ with $n$ instances $x$ and labels $y$
$r \leftarrow$    max boosting rounds
$a \leftarrow 0$    All coefficients are 0
$\mathcal{H}(S, u)$    Weak learner
$u \leftarrow (\frac{1}{n}, \ldots, \frac{1}{n})$  Example weights
for $j$ = 1..$r$
  Find weak learner  $h_j \leftarrow \mathcal{H}(S, u)$
  $\epsilon_j \leftarrow \sum_{i:h_j(x_i) \neq y_i} u_i$    weighted error
  if $\epsilon_j > 1/2, r \leftarrow j - 1$  break
  $a_j \leftarrow \log \frac{\epsilon_j}{1-\epsilon_j}$    hypothesis weight
  for each $u_i$
     if $h_j(x_i) \neq y_i, u_i \leftarrow u_i/(2\epsilon_j)$
     else,              $u_i \leftarrow u_i/(2(1 - \epsilon_j))$
  end
end
return $r, f = \sum_{j=1}^{r} a_j h_j$
```

The basic rationale behind AdaBoost is to increase the weight values of the misclassified objects and to decrease the weight values of those classified correctly in each iteration. Thus, the next weak learner attempts to perform well on the misclassified objects. Although, this black-box method might look to be prone to the overfitting problem in each iteration, strong experimental results have shown that AdaBoost can generalize very well even when the number of boosting rounds is increased. Breiman tried to explain this situation in the context of Bias-Variance trade-off [22]. It was shown [87] that better generalization was due to the maximization of the margin distribution in label space ($yf(x)$).

Margin maximization in label space enables the determination of the generalization error boundaries of boosting approach by adapting similar boundaries from support vector machines. Such error boundaries, dependent on the function class, the size of the smallest cover for such function class, and the size of the training set are given in [10, 40] (especially Theorem 2.2 of [40]). We give this theorem without proof.

Theorem 5.2.1. *Consider thresholding a real-valued function space $\mathcal{F}$ on the domain X. Fix $\gamma \in \mathbb{R}^+$ and choose $\mathcal{G} \subset \mathcal{F} \times L(X)$. For any probability distribution $\mathcal{D}$ on $X \times \{-1, 1\}$, with probability $1-\delta$ over n random examples S, any hypothesis $f \in \mathcal{F}$ for which $(f, g_f) \in \mathcal{G}$ has generalization error no more than*

$$\mathrm{err}_{\mathcal{D}}(f) \leq \varepsilon(n, \mathcal{F}, \delta, \gamma) = \frac{2}{n}\left(\log \mathcal{N}(\mathcal{G}, 2n, \frac{\gamma}{2}) + \log \frac{2}{\delta}\right),$$

where $\mathcal{N}(\cdot)$ is the covering number and provided $n > 2/\varepsilon$, and there is no discrete probability on misclassified training points.

We are now in a position to apply these results to our function class which will be in the form described above, $\mathcal{F} = \mathrm{co}(H) = \left\{\sum_{h \in H} a_h h : a_h \geq 0\right\}$, where we have left open for the time being what the class H of weak learners might contain. The sets $\mathcal{G}$ of Theorem 5.2.1 will be chosen as follows:

$$\mathcal{G}_B = \left\{\left(\sum_{h \in H} a_h h, g\right) : \sum_{h \in H} a_h + \|g\|_1 \leq B,\ a_h \geq 0\right\}.$$

Hence, the condition that a function $f = \sum_{h \in H} a_h h$ satisfies the conditions of Theorem 5.2.1 for $\mathcal{G} = \mathcal{G}_B$ is simply

$$\begin{aligned} &\textstyle\sum_{h \in H} a_h + \frac{1}{\Delta}\sum_{i=1}^{n} \xi\left((\mathbf{x}_i, y_i), f, \gamma\right) \\ &\quad = \textstyle\sum_{h \in H} a_h + \frac{1}{\Delta}\sum_{i=1}^{n} \xi_i \leq B. \end{aligned} \tag{5.1}$$

Note that this will be the quantity that we will minimize through the boosting iterations described in later sections, where we will use the parameter C in place of $1/\Delta$ and the margin γ will be set to 1. Based on results from [40], we see that optimizing B directly optimizes the relevant covering number bound and hence the generalization bound given in Theorem 5.2.1 with $\mathcal{G} = \mathcal{G}_B$. Note that in the cases considered, $|G|$ is just the growth function, $B_H(m)$, of the class, H of weak learners. The central focus of this chapter is to optimize these error boundaries using a column generation technique from mathematical programming. In the following sections specific formulations based on linear programming are introduced.

5.3 Boosting LP for Classification

From the theoretical results shown in [10, 40], we can see that a soft margin cost function should be valuable for boosting classification functions. Once again using the techniques used in support vector machines, we can formulate this problem as a linear program. The upper bound on generalization error defined in [40] can be optimized directly using an LP. The LP is formulated as if all possible labelings of the training data by the weak learners were known. The LP minimizes the 1-norm soft margin cost function used in support vector machines with the added restrictions that all the weights are positive and the threshold is assumed to be zero. This LP and variants can be practically solved using a column generation approach. Weak learners are generated as needed to produce the optimal support vector machine based on the output of the all weak learners. In essence the base learner become an 'oracle' that generates the necessary columns. The dual variables of the linear program provide the misclassification costs needed by the learning machine. The column generation procedure searches for the best possible misclassification costs in dual space. Only at optimality is the actual ensemble of weak learners constructed.

5.3.1 LP Formulation

Let the matrix H be a n by r matrix of all the possible labelings of the training data using functions from $\mathcal{H}$. Specifically $H_{ij} = h_j(x_i)$ is the label $(1 \; or -1)$ given by weak learner $h_j \in \mathcal{H}$ on the training point x_i. Each column $H_{.j}$ of the matrix H constitutes the output of weak learner $h_j \;\; j = 1, \ldots, r$ on the training data, while each row H_i gives the outputs of all the weak learners on the example $x_i \;\; i = 1, \ldots, n$. There may be up to 2^n distinct weak learners.

The following linear program can be used to minimize upper bound on generalization error (B) given in Eq.5.1:

$$\begin{array}{ll} \min_{a,\xi} & \sum_{i=1}^{r} a_i + C\sum_{i=1}^{n} \xi_i \\ s.t. & y_i H_i a + \xi_i \geq 1, \quad \xi_i \geq 0, \ i = 1, \ldots, n \\ & a_j \geq 0, \ i = 1, \ldots, r \end{array} \tag{5.2}$$

where $C > 0$ is the tradeoff parameter between misclassification error and margin maximization. The dual of LP (5.2) is

$$\begin{array}{ll} \max_{u} & \sum_{i=1}^{n} u_i \\ s.t. & \sum_{i=1}^{n} u_i y_i H_{ij} \leq 1, \ j = 1, \ldots, r \\ & 0 \leq u_i \leq C, \ i = 1, \ldots, n \end{array} \tag{5.3}$$

Alternative soft margin LP formulations exist, such as this one for the ν-LP Boosting[4]. [81]:

$$\begin{array}{ll} \max_{a,\xi,\rho} & \rho - D\sum_{i=1}^{n} \xi_i \\ s.t. & y_i H_i a + \xi_i \geq \rho, \ i = 1, \ldots, n \\ & \sum_{i=1}^{r} a_i = 1, \quad \xi_i \geq 0, \ i = 1, \ldots, n \\ & a_j \geq 0, \ j = 1, \ldots, r \end{array} \tag{5.4}$$

The dual of this LP (5.4) is:

$$\begin{array}{ll} \min_{u,\beta} & \beta \\ s.t. & \sum_{i=1}^{n} u_i y_i H_{ij} \leq \beta, \ j = 1, \ldots, r \\ & \sum_{i=1}^{n} u_i = 1, \quad 0 \leq u_i \leq D, \ i = 1, \ldots, n \end{array} \tag{5.5}$$

These LP formulations are exactly equivalent given the appropriate choice of the parameters C and D. Proofs of this fact can be found in [82, 10] so we only state the theorem here.

Theorem 5.3.1 (LP Formulation Equivalence). *If LP (5.4) with parameter D has a primal solution* $(\bar{a}, \bar{\rho} > 0, \bar{\xi})$ *and dual solution* $(\bar{u}, \bar{\beta})$, *then* $(\hat{a} = \frac{\bar{a}}{\bar{\rho}}, \hat{\xi} = \frac{\bar{\xi}}{\bar{\rho}})$

[4]We remove the constraint $\rho \geq 0$ since $\rho > 0$ at optimality under the complementation assumption.

and $(\hat{u} = \frac{\bar{u}}{\hat{\beta}})$ *are the primal and dual solutions of LP (5.2) with parameter* $C = \frac{D}{\hat{\beta}}$*. Similarly, if LP 5.2 with parameter C has primal solution* $(\hat{a} \neq 0, \hat{\xi})$ *and dual solution* $(\hat{u} \neq 0)$*, then* $(\bar{\rho} = \frac{1}{\sum_{i=1}^{r} \hat{a}_i}, \bar{a} = \hat{a}\bar{\rho}, \bar{\xi} = \hat{\xi}\bar{\rho})$ *and* $(\bar{\beta} = \frac{1}{\sum_{i=1}^{n} \hat{u}_i}, \bar{u} = \hat{u}\bar{\beta})$ *are the primal and dual solutions of LP (5.4) with parameter* $D = C\hat{\beta}$*.*

Practically we found ν-LP (5.4) with $D = \frac{1}{n\nu}$, $\nu \in (0,1)$ preferable because of the interpretability of the parameter. A more extensive discussion and development of these characteristics for SVM classification can be found in [82]. To maintain dual feasibility, the parameter ν must maintain $\frac{1}{n} <= D <= 1$. By picking ν appropriately we can force the minimum number of support vectors. We know that the number of support vectors will be the number of points misclassified plus the points on the margin, and this was used as a heuristic for choice of ν. The reader should consult [81, 82] for a more in-depth analysis of this family of cost functions.

5.3.2 Properties of LP formulation

We now examine the characteristics of LP (5.4) and its optimality conditions to gain insight into the properties of LP Boosting. This will be useful in understanding both the effects of the choice of parameters in the model and the performance of the eventual algorithm. The optimality conditions [72] of LP (5.4) are *primal feasibility*:

$$\begin{array}{l} y_i H_i a + \xi_i \geq \rho, \;\; i = 1, \ldots, n \\ \sum_{i=1}^{r} a_i = 1, \;\;\; \xi_i \geq 0, \;\; i = 1, \ldots, n \\ a_j \geq 0, \;\; i = 1, \ldots, r \end{array} \tag{5.6}$$

dual feasibility:

$$\begin{array}{l} \sum_{i=1}^{n} u_i y_i H_{ij} \leq \beta, \;\; j = 1, \ldots, r \\ \sum_{i=1}^{n} u_i = 1, \;\;\; 0 \leq u_i \leq D, \;\; i = 1, \ldots, n \end{array} \tag{5.7}$$

and *complementarity* here stated as equality of the primal and dual objectives:

$$\rho - D \sum_{i=1}^{n} \xi_i = \beta \tag{5.8}$$

Complementarity can be expressed using many equivalent formulations. For example, from the complementarity property, the following equations hold:

$$\begin{aligned} &u_i(y_i H_i a + \xi_i - \rho) = 0, \ i = 1, \ldots, n \\ &a_j(\textstyle\sum_{i=1}^n u_i y_i H_{ij} - \beta) = 0, \ j = 1, \ldots, r \end{aligned} \tag{5.9}$$

As in SVM, the optimality conditions tell us many things. First we can characterize the set of base learners that are positively weighted in the optimal ensemble. Recall that the primal variables a_i multiply each base learner. The dual LP assigns misclassification costs u_i to each point such that the u_i sum to 1. The dual constraint $\sum_{i=1}^n u_i y_i H_{ij} \leq \beta$ "scores" each weak learner $h_{.j}$. The score is the weighted sum of the correctly classified points minus the weighted sum of the incorrectly classified points. The weak learners with lower scores have greater weighted misclassification costs. The formulation is pessimistic in some sense. The set of best weak learners for a given u will all have a score of β. The dual objective minimizes β so the optimal misclassification cost u will be the most pessimistic one, i.e., it minimizes the maximum score over all the weak learners. From the complementary slackness condition, $a_j(\sum_{i=1}^n u_i y_i H_{ij} - \beta) = 0, \ j = 1, \ldots, r$, only the weak learners with scores equal to β can have positive weights a_j in the primal space. So the resulting ensemble will be a linear combination of the weak learners that perform best under the most pessimistic choice of misclassification costs. This interpretation closely corresponds to the game strategy approach of [23] (which is also a LP boosting formulation solvable by LPBoost.) A notable difference is that LP (5.5) has an additional upper bound on the misclassification costs u, $0 \leq u_i \leq D, \ i = 1, \ldots, n$, that is produced by the introduction of the soft margin in the primal.

From SVM research, we know that both the primal and dual solutions will be sparse and the degree of sparsity will be greatly influenced by the choice of parameter $D = \frac{1}{\nu n}$. The size of the dual feasible region depends on our choice of ν. If ν is too large, forcing D small, then the dual problem is infeasible. For large but still feasible ν (D very small but still feasible), the problem degrades to something very close to the equal-cost case, $u_i = 1/n$. All the u_i are forced to be nonzero. Practically, this means that as ν increases, the optimal solution is frequently a single weak learner that is best assuming equal costs. As ν decreases (D grows), the misclassification costs, u_i, will increase for hard-to-classify points or points on the margin in the label space and will go to 0 for points that are easy to classify. Thus the misclassification costs u become sparser. If ν is too small (and D too large) then the meaningless null solution, $a = 0$, with all points classified as one class, becomes optimal.

For a good choice of ν, a sparse solution for the primal ensemble weights a will be optimal. This implies that few weak learners will be used. Also a sparse dual u will be optimal. This means that the solution will be dependent only on a smaller subset of data (the support vectors.) Data with $u_i = 0$ are well-classified with sufficient margin, so the performance on these data is not critical. From LP sensitivity analysis, we know that the u_i are exactly the sensitivity of the optimal solution to small perturbations in the margin. In some sense the sparseness of u is good because the weak learners can be constructed using only smaller subsets of the data. But as we will see in Section 5.7, this sparseness of the misclassification costs can lead to problems when practically implementing algorithms.

5.4 LPBoost Algorithms

We now examine practical algorithms for solving the LP (5.4). Since the matrix H has a very large number of columns, prior authors have dismissed the idea of solving LP formulations for boosting as being intractable using standard LP techniques. But column generation techniques for solving such LPs have existed since the 1950s and can be found in LP text books; see for example [72, Section 7.4]. Column generation is frequently used in large-scale integer and linear programming algorithms so commercial codes such as CPLEX have been optimized to perform column generation very efficiently [33]. The simplex method does not require that the matrix H be explicitly available. At each iteration, only a subset of the columns is used to determine the current solution (called a basic feasible solution). The simplex method needs some means for determining if the current solution is optimal, and if it is not, some means for generating some column that violates the optimality conditions. The tasks of verification of optimality and generating a column can be performed by the learning algorithm. A simplex-based boosting method will alternate between solving an LP for a reduced matrix $\hat{H}$ corresponding to the weak learners generated so far and using the weak learning algorithm to generate the best-scoring weak learner based on the dual misclassification cost provided by the LP. This will continue until a well-defined exact or approximate stopping criterion is reached.

The idea of column generation (CG) is to restrict the primal problem (5.2) by considering only a subset of all the possible labelings based on the weak learners generated so far; i.e., only a subset $\hat{H}$ of the columns of H is used. The LP solved using $\hat{H}$ is typically referred to as the *restricted master problem.* Solving the restricted primal LP corresponds to solving a relaxation of the dual LP. The constraints for weak learners that have not been generated yet are missing. One extreme case is when no weak learners are considered. In this case the optimal dual solution is $\hat{u}_i = \frac{1}{n}$ (with appropriate choice of D). This will provide the initialization of the algorithm.

If we consider the unused columns to have $\hat{a}_i = 0$, then $\hat{a}$ is feasible for the original primal LP. If $(\hat{u}, \hat{\beta})$ is feasible for the original dual problem then we are done since we have primal and dual feasibility with equal objectives. If $\hat{a}$ is not optimal then $(\hat{u}, \hat{\beta})$ is infeasible for the dual LP with full matrix H. Specifically, the constraint $\sum_{i=1}^{n} \hat{u}_i y_i H_{ij} \leq \hat{\beta}$ is violated for at least one weak learner. Or equivalently, $\sum_{i=1}^{n} \hat{u}_i y_i H_{ij} > \hat{\beta}$ for some j. Of course we do not want to a priori generate all columns of H $(H_{.j})$, so we use our weak learner as an oracle that either produces $H_{.j}$, $\sum_{i=1}^{n} \hat{u}_i y_i H_{ij} > \hat{\beta}$ for some j, or a guarantee that no such $H_{.j}$ exists. To speed convergence we would like to find the one with maximum deviation, that is, the weak learning algorithm $\mathcal{H}(S, u)$ must deliver a function $\hat{h}$ satisfying

$$\sum_{i=1}^{n} y_i \hat{h}(x_i) \hat{u}_i = \max_{h \in \mathcal{H}} \sum_{i=1}^{n} \hat{u}_i y_i h(x_i) \tag{5.10}$$

Thus $\hat{u}_i$ becomes the new misclassification cost, for example i, that is given to the weak learning machine to guide the choice of the next weak learner. One of the big payoffs of the approach is that we have a stopping criterion. If there is no weak learner h for which $\sum_{i=1}^{n} \hat{u}_i y_i h(x_i) > \hat{\beta}$, then the current combined hypothesis is the optimal solution over all linear combinations of weak learners.

We can also gauge the cost of early stopping since if $\max_{h \in \mathcal{H}} \sum_{i=1}^{n} \hat{u}_i y_i h(x_i)) \leq \hat{\beta} + \epsilon$, for some $\epsilon > 0$, we can obtain a feasible solution of the full dual problem by taking $(\hat{u}, \hat{\beta} + \epsilon)$. Hence, the value V of the optimal solution can be bounded between $\hat{\beta} \leq V < \hat{\beta} + \epsilon$. This implies that, even if we were to potentially include a non-zero coefficient for all the weak learners, the value of the objective $\rho - D \sum_{i=1}^{n} \xi_i$ can only be increased by at most ϵ.

We assume the existence of the weak learning algorithm $\mathcal{H}(S, u)$ which selects the best weak learner from a set H closed under complementation using the criterion of equation (5.10). The following algorithm results

Algorithm 5.4.1 (LPBoost).

```
$Given\ as\ input\ training\ set:\ S$
$r \leftarrow 0$   $No\ weak\ learners$
$a \leftarrow 0$   $All\ coefficients\ are\ 0$
$\beta \leftarrow 0$
$u \leftarrow (\frac{1}{n}, \dots, \frac{1}{n})$   $Corresponding\ optimal\ dual$
$REPEAT$
  $r \leftarrow r + 1$
  $Find\ weak\ learner\ using\ equation\ (5.10):$
  $h_r \leftarrow \mathcal{H}(S, u)$
  $Check\ for\ optimal\ solution:$
  $If \sum_{i=1}^{n} u_i y_i h_r(x_i) \leq \beta, r \leftarrow r - 1, break$
  $H_{ir} \leftarrow h_r(x_i)$
  $Solve\ restricted\ master\ for\ new\ costs:$
                    $argmin$   $\beta$
  $(u, \beta) \leftarrow$   $s.t.$   $\sum_{i=1}^{n} u_i y_i h_j(x_i) \leq \beta$
                               $j = 1, \dots, r$
                               $0 \leq u_i \leq D,\ i = 1, \dots, n$
$END$
$a \leftarrow Lagrangian\ multipliers\ from\ last\ LP$
$return\ r, f = \sum_{j=1}^{r} a_j h_j$
```

Note that the assumption of finding the best weak learner is not essential for good performance on the algorithm. Recall that the role of the learning algorithm is to generate columns (weak learners) corresponding to a dual infeasible row or to indicate optimality by showing no infeasible weak learners exist. All that we require is that the base learner return a column corresponding to a dual infeasible row. It need not be the one with maximum infeasibility. This is merely done to improve convergence speed. In fact, choosing columns using "steepest edge" criteria that look for the column that leads to the biggest actual change in the objective may lead to even faster convergence. If the learning algorithm fails to find a dual infeasible weak learner when one exists than the algorithm may prematurely stop at a nonoptimal solution.

With small changes this algorithm can be adapted to perform any of the LP boosting formulations by simply changing the restricted master LP solved, the costs given to the learning algorithm, and the optimality conditions checked. Assuming the base learner solves (5.10) exactly, LPBoost is a variant of the dual simplex algorithm [72]. Thus it inherits all the benefits of the simplex algorithm. Benefits include:

1. Well-defined exact and approximate stopping criteria. Typically, ad hoc termination schemes, e.g. a fixed number of iterations, must be used for the gradient-based boosting algorithms.

2. Finite termination at a globally optimal solution. In practice the algorithm generates few weak learners to arrive at an optimal solution.

3. The optimal solution is sparse and thus uses few weak learners.

4. The algorithm is performed in the dual space of the classification costs. The weights of the optimal ensemble are only generated and fixed at optimality.

5. High-performance commercial LP algorithms optimized for column generation exist that do not suffer from the numeric instability problems reported for boosting [2].

5.5 Confidence-rated Boosting

The derivations and algorithm of the last two sections did not rely on the assumption that $H_{ij} \in \{-1, +1\}$. We can therefore apply the same reasoning to implementing a weak learning algorithm for a finite set of confidence-rated functions $\mathcal{F}$ whose outputs are real numbers. We again assume that $\mathcal{F}$ is closed under complementation. We simply define $H_{ij} = f_j(x_i)$ for each $f_j \in \mathcal{F}$ and apply the same algorithm as before. We again assume the existence of a weak learner $F(S, u)$, which finds a function $\hat{f} \in \mathcal{F}$ satisfying

$$\sum_{i=1}^{n} y_i \hat{f}(x_i)\hat{u}_i = \max_{f \in \mathcal{F}} \sum_{i=1}^{n} \hat{u}_i y_i f(x_i) \tag{5.11}$$

The only difference in the associated algorithm is the weak learner which now optimizes this equation.

Algorithm 5.5.1 (LPBoost-CRB).

```
Given as input training set: $S$
$r \leftarrow 0$   No weak learners
$a \leftarrow 0$   All coefficients are $0$
$\beta \leftarrow 0$
$u \leftarrow (\frac{1}{n}, \dots, \frac{1}{n})$   Corresponding optimal dual
REPEAT
  $r \leftarrow r + 1$
  Find weak learner using equation (5.11) :
  $f_r \leftarrow \mathcal{F}(S, u)$
  Check for optimal solution:
  If $\sum_{i=1}^{n} u_i f_i h_r(x_i) \le \beta, r \leftarrow r - 1$, break
  $H_{ir} \leftarrow f_r(x_i)$
  Solve restricted master for new costs:
  $(u, \beta) \leftarrow \begin{array}{ll} \text{argmin} & \beta \\ \text{s.t.} & \sum_{i=1}^{n} u_i y_i f_j(x_i) \le \beta \\ & j = 1, \dots, r \\ & 0 \le u_i \le D,\ i = 1, \dots, n \end{array}$
END
$a \leftarrow$ Lagrangian multipliers from last LP
return $r, f = \sum_{j=1}^{r} a_j f_j$
```

5.6 LPBoost for Regression

The LPBoost algorithm can be extended to optimize any ensemble cost function that can be formulated as a linear program. To solve alternate formulations we need only change the LP restricted master problem solved at each iteration and the criteria given to the base learner. The only assumptions in the current approach are that the number of weak learners be finite and that if an improving weak learner exists then the base learner can generate it. To see a simple example of this consider the problem of boosting regression functions. We use the following adaptation of the SVM regression formulations. This LP was also adapted to boosting using a barrier algorithm in [80]. We assume we are given a training set of data $S = ((\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n))$, but now y_i may take on any real value.

$$\begin{array}{lll} \min_{a,\xi,\xi^*,\epsilon} & C\nu\epsilon + \eta\sum_{i=1}^{r} a_i + C\sum_{i=1}^{n}(\xi_i + \xi_i^*) & \\ s.t. & H_i a - y_i - \xi_i \leq \epsilon, & \xi_i \geq 0,\ i = 1, \ldots, n \\ & H_i a - y_i + \xi_i^* \geq -\epsilon, & \xi_i^* \geq 0,\ i = 1, \ldots, n \\ & a_j \geq 0,\ i = 1, \ldots, r & \end{array} \tag{5.12}$$

First we reformulate the problem slightly differently:

$$\begin{array}{lll} \min_{a,\xi,\xi^*,\epsilon} & C\nu\epsilon + \eta\sum_{i=1}^{r} a_i + C\sum_{i=1}^{n}(\xi_i + \xi_i^*) & \\ s.t. & \epsilon - H_i a + \xi_i \geq -y_i, & \xi_i \geq 0,\ i = 1, \ldots, n \\ & \epsilon + H_i a + \xi_i^* \geq y_i, & \xi_i^* \geq 0,\ i = 1, \ldots, n \\ & a_i \geq 0,\ i = 1, \ldots, r & \end{array} \tag{5.13}$$

We introduce Lagrangian multipliers (u, u^*), construct the dual, and convert to a minimization problem to yield:

$$\begin{array}{ll} \min_{u,u^*} & \sum_{i=1}^{n} y_i(u_i - u_i^*) \\ s.t. & \sum_{i=1}^{n}(-u_i + u_i^*)H_{ij} \leq \eta,\ j = 1, \ldots, r \\ & \sum_{i=1}^{n}(u_i + u_i^*) = 1 \\ & 0 \leq u_i \leq C,\quad 0 \leq u_i^* \leq C,\ i = 1, \ldots, n \end{array} \tag{5.14}$$

LP (5.14) restricted to all weak learners constructed so far becomes the new master problem. If the base learner returns any hypothesis $H_{.j}$ that is not dual feasible, i.e. $(\sum_{i=1}^{n}(-u_i + u_i^*)H_{ij} > \eta)$, then the ensemble is not optimal and the weak learner should be added to the ensemble. To speed convergence we would like the weak learner with maximum deviation, i.e.,

$$max_j \sum_{i=1}^{n}(-u_i + u_i^*)H_{ij}. \tag{5.15}$$

This is perhaps odd at first glance because the criteria do not actually explicitly involve the dependent variables y_i. But within the LPBoost algorithm, the u_i are closely related to the error residuals of the current ensemble. If the data point x_i is overestimated by the current ensemble function by more than ϵ, then by complementarity u_i will be positive and $u_i^* = 0$. So at the next iteration the weak learner will attempt to construct a function that has a negative sign at point x_i. If the point x_i falls within the ϵ margin then the $u_i = u_i^* = 0$, and the next weak learner will try to construct a function with value 0 at that point. If the data point x_i is underestimated by the current ensemble function by more than ϵ, then by complementarity u_i^* will be positive and $u_i = 0$. So at the next iteration the weak learner will attempt to construct a function that has a positive sign at point x_i. By sensitivity analysis, the magnitudes of u and u^* are proportional to the changes of the objective with respect to changes in the margin.

This becomes even clearer using the approach taken in the Barrier Boosting algorithm for this problem [80]. Equation (5.15) can be converted to a least squares problem. For $v_i = -u_i + u_i^*$ and $H_{ij} = f_j(x_i)$,

$$(f(x_i) - v_i)^2 = f(x_i)^2 - 2v_i f(x_i) + v_i^2. \tag{5.16}$$

So the objective to be optimized by the weak learner can be transformed as follows:

$$\begin{aligned} max_j \sum_{i=1}^{n}(-u_i + u_i^*)f_j(x_i) &= max_j \sum_{i=1}^{n} v_i f_j(x_i) \qquad (5.17)\\ &= \frac{-1}{2} min_j \sum_{i=1}^{n}\left[(f_j(x_i) - v_i)^2 - f_j(x_i)^2 - v_i^2\right]. \end{aligned}$$

The constant term v_i^2 can be ignored. So effectively the weak learner must construct a regularized least squares approximation of the residual function.

The final regression algorithm looks very much like the classification case. The variables u_i and u_i^* can be initialized to any initial feasible point. We present one such strategy here assuming that D is sufficiently large. Here $(a)_+ := \max(a, 0)$ denotes the plus function.

Algorithm 5.6.1 (LPBoost-Regression).

```
Given as input training set: $S$
$r \leftarrow 0$   No weak learners
$a \leftarrow 0$   All coefficients are $0$
$u_i \leftarrow \frac{(-y_i)_+}{||y||_1}$   Corresponding feasible dual
$u_i^* \leftarrow \frac{(y_i)_+}{||y||_1}$
REPEAT
  $r \leftarrow r + 1$
  Find weak learner using equation (5.17):
  $h_r \leftarrow \mathcal{H}(S, (-u + u^*))$
  Check for optimal solution:
  If $\sum_{i=1}^{n}(-u_i + u_i^*)h_r(x_i) \leq \eta, r \leftarrow r - 1,$ break
  $H_{ir} \leftarrow h_r(x_i)$
  Solve restricted master for new costs:
                        argmin  $\sum_{i=1}^{n}(u_i - u_i^*)y_i$
                        s.t.    $\sum_{i=1}^{n}(-u_i + u_i^*)h_j(x_i) \leq \eta$
  $(u, u^*) \leftarrow$                 $j = 1, \dots, r$
                                $\sum_{i=1}^{n}(u_i + u_i^*) = 1$
                                $0 \leq u_i, u_i^* \leq C,\ i = 1, \dots, n$
END
$a \leftarrow$ Lagrangian multipliers from last LP
return $r, f = \sum_{j=1}^{r} a_j h_j$
```

5.7 Computational Experiments

We performed three sets of experiments to compare the performance of LPBoost, CRB, and AdaBoost on three classification tasks: one boosting decision tree stumps on smaller datasets and two boosting C4.5 [79]. For decision tree stumps

Table 5.1: Average Accuracy and Standard Deviations of Boosting using Decision Tree Stumps (r) = number of stumps in final ensemble

Dataset	LPBoost (r)	AB-100	AB-1000
Cancer	0.9657 ± 0.0245 (14.7)	0.9542 ± 0.0292	0.9471 ± 0.0261
Diagnostic	0.9613 ± 0.0272 (54.2)	0.9684 ± 0.0273	0.9701 ± 0.0311
Heart	0.7946 ± 0.0786 (70.8)	0.8182 ± 0.0753	0.8014 ± 0.0610
Ionosphere	0.9060 ± 0.0523 (87.6)	0.9060 ± 0.0541	0.9031 ± 0.0432
Musk	0.8824 ± 0.0347 (205.3)	0.8403 ± 0.0415	0.8908 ± 0.0326
Sonar	0.8702 ± 0.0817 (85.7)	0.8077 ± 0.0844	0.8558 ± 0.0781

six datasets were used. For the C4.5 experiments, we report results for four large datasets with and without noise. Finally, to further validate C4.5, we experimented with ten more additional datasets. The rationale was to first evaluate LPBoost where the base learner solves (5.10) exactly, then to examine LPBoost in a more realistic environment by using C4.5 as a base learner. All of the datasets were obtained from the UC-Irvine data repository [68]. For the C4.5 experiments we performed both traditional and confidence- rated boosting.

5.7.1 Boosting Decision Tree Stumps

We used decision tree stumps as a base learner on the following six datasets: Cancer (9,699), Diagnostic (30,569), Heart (13,297), Ionosphere (34,351), Musk (166,476), and Sonar (60,208). The number of features and number of points in each dataset are shown, respectively, in parentheses. We report testing set accuracy for each dataset based on 10-fold Cross Validation (CV). We generate the decision tree stumps based on the mid-point between two consecutive points for a given variable. Since there is limited confidence information in stumps, we did not perform confidence-rated boosting. All boosting methods search for the best weak learner which returns the least weighted misclassification error at each iteration. LPBoost can take advantage of the fact that each weak learner need only be added into the ensemble once. Thus once a stump is added to the ensemble it is never evaluated by the learning algorithm again. The weights of the weak learners are adjusted dynamically by the LP. This is an advantage over AdaBoost, since AdaBoost adjusts prior weights by repeatedly adding the same weak learner into the ensemble.

The parameter ν for LPBoost was set using a simple heuristic: 0.1 added to previously-reported error rates on each dataset in [9] except for the Cancer dataset. Specifically the values of ν in the same order of the datasets given above were (0.2, 0.1, 0.25, 0.2, 0.25, 0.3). Results for AdaBoost were reported for a maximum number of iterations of 100 and 1000. The 10-fold average classification accuracies and standard deviations are reported in Table 5.1.

LPBoost performed very well both in terms of classification accuracy, number of weak learners, and training time. There is little difference between the accuracy of LPBoost and the best accuracy reported for AdaBoost using either 100 or 1000 iterations. The variation in AdaBoost for 100 and 1000 iterations illustrates the importance of well-defined stopping criteria. Typically, AdaBoost only obtains its solution in the limit and thus stops when the maximum number of iterations (or some other heuristic stopping criteria) is reached. There is no magic number of iterations good for all datasets. LPBoost has a well-defined stopping criterion that is reached in a few iterations. It uses few weak learners. There are only 81 possible stumps on the Breast Cancer dataset (nine attributes having nine possible values), so clearly AdaBoost may require the same tree to be generated multiple times. LPBoost generates a weak learner only once and can alter the weight on that weak learner at any iteration. The run time of LPBoost is proportional to the number of weak learners generated. Since the LP package that we used, CPLEX 4.0 [33], is optimized for column generation, the cost of adding a column and reoptimizing the LP at each iteration is small. An iteration of LPBoost is only slightly more expensive that an iteration of AdaBoost. The time is proportional to the number of weak learners generated. For problems in which LPBoost generates far fewer weak learners it is much less computationally costly.

In the next subsection, we test the practicality of our methodology on different datasets using C4.5.

5.7.2 Boosting C4.5

LPBoost with C4.5 as the base algorithm performed well after some operational challenges were solved. In concept, boosting using C4.5 is straightforward since the C4.5 algorithm accepts misclassification costs. One problem is that C4.5 only finds a good solution not guaranteed to maximize (5.10). This can effect the convergence speed of the algorithm and may cause the algorithm to terminate at a suboptimal solution. Another challenge is that the misclassification costs determined by LPBoost are sparse, i.e. $u_i = 0$ for many of the points. The dual LP has a basic feasible solution corresponding to a vertex of the dual feasible region. Only the variables corresponding to the basic solution can be nonnegative. So while a face of the region corresponding to many nonnegative weights may be optimal, only a vertex solution will be chosen. In practice we found that when many $u_i = 0$, LPBoost converged slowly. In the limited number of iterations that we allowed (25), LPBoost frequently failed to find weak learners that improved significantly over the initial equal cost solution. The weak learners generated using only subsets of the variables were not necessarily good over the full data set. Thus the search was too slow. Alternative optimization algorithms may alleviate this problem. For example, an interior point strategy may lead to significant performance improvements. Note that other authors have reported problems with underflow of boosting [2]. When LPBoost was solved to optimality on decision tree stumps with full evaluation of the weak learners, this problem did not occur. Boosting unpruned decision trees helped somewhat but did not completely eliminate this problem.

Stability and convergence speed was greatly improved by adding minimum misclassification costs to the dual LP (5.5) :

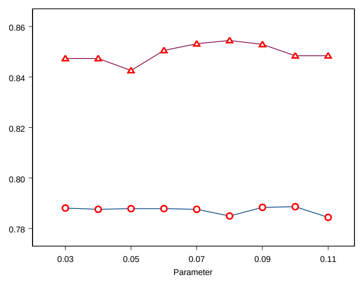
$$\begin{array}{ll} \min_u & \beta \\ s.t. & \sum_{i=1}^n u_i y_i H_{ij} \leq \beta, \ j = 1, \dots, r \\ & \sum_{i=1}^n u_i = 1 \\ & D' \leq u_i \leq D, \ i = 1, \dots, n \end{array} \tag{5.18}$$

where $D = \frac{1}{\nu n}$ and $D' = \frac{1}{25\nu n}$. The corresponding primal problem is

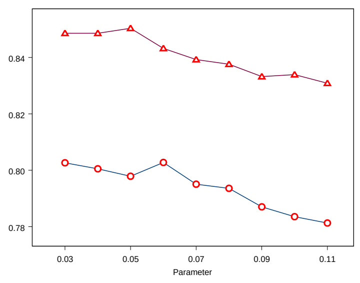
$$\begin{array}{ll} \max_{a,\xi,\rho} & \rho + D' \sum_{i=1}^{n} \tau_i - D \sum_{i=1}^{n} \xi_i \\ s.t. & y_i H_i a + \xi_i \geq \rho + \tau_i, \ i = 1, \dots, n \\ & \sum_{i=1}^{r} a_i = 1, \ , a_j \geq 0, \ i = 1, \dots, r \\ & \xi_i \geq 0, \ i = 1, \dots, n \end{array} \tag{5.19}$$

The primal problem maximizes two measures of soft margin: ρ corresponds to the minimum margin obtained by all points and τ_i measures the additional margin obtained by each point. AdaBoost also minimizes a margin cost function based on the margin obtained by each point.

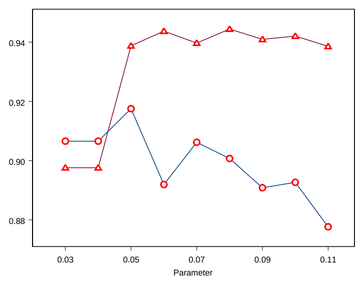
We ran experiments on larger datasets: Forest, Adult, USPS, and Optdigits from UCI[68]. LPBoost was adopted to the multiclass problem by defining $h_j(x_i) = 1$ if instance x_i is correctly classified by weak learner h_j and -1 otherwise.This is just one method of boosting multiclass problems. Further investigation of multiclass approaches is needed. Forest is a 54-dimension dataset with seven possible classes. The data are divided into 11340 training, 3780 validation, and 565892 testing instances. There are no missing values. The 15-dimensional Adult dataset has 32562 training and 16283 testing instances. One training point that has a missing value for a class label has been removed. We use 8140 instances as our training set and the remaining 24421 instances as the validation set. Adult is a two-class dataset with missing values. The default handling in C4.5 has been used for missing values. USPS and Optdigits are optical character recognition datasets. USPS has 256 dimensions without missing value. Out of 7291 original training points, we use 1822 points as training data and the rest 5469 as validation data. There are 2007 test points. Optdigits on the other hand has 64 dimensions without missing values. Its original training set has 3823 points. We use 955 of them as training data and the remaining 2868 as validation data. Parameter selection for both LPBoost and AdaBoost was done based on validation set results. Since initial experiments resulted in the same parameter set for both LPBoost and CRB, we set the parameters equal for CRB and LPBoost to expedite computational work. In order to investigate the performance of boosted C4.5 with noisy data, we introduced 15% label noise for all



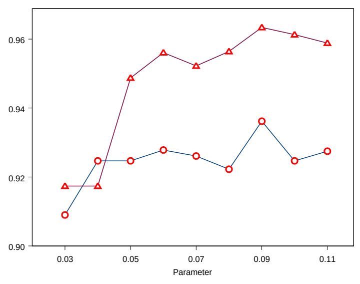
(a) Forest Dataset



(b) Adult Dataset



(c) USPS Dataset



(d) Optdigits Dataset

Figure 5.1: Validation Set Accuracy by ν Value. Triangles are no noise and circles are with noise.

four datasets.

The ν parameter used in LPBoost and the number of iterations of AdaBoost can significantly affect their performance. Thus accuracy on the validation set was used to pick the parameter ν for LPBoost and the number of iterations for AdaBoost. To avoid excessive computation, we limit the maximum number of iterations at 25 for all boosting methods as in [2]. We varied parameter ν between 0.03 and 0.11. Initial experiments indicated that for very small ν values, LPBoost results in one classifier which assigns all training points to one class. On the other extreme, for larger values of ν, LPBoost returns one classifier which is equal to the one found in the first iteration. Figure 5.1 shows the validation set accuracy for LPBoost on all four datasets. Based on validation set results, we use (22,19), (25,4), (22,25), and (25,25) number of iterations for original and 15% noisy data respectively for AdaBoost in the Forest, Adult, USPS, and Optdigits datasets.

Table 5.2: Large Dataset Results from Boosting C4.5

Dataset	LPBoost	CRB	AdaBoost	C4.5
Original Forest	0.7226	0.7259	**0.7370**	0.6638
+ 15% Noise	0.6602	0.6569	**0.6763**	0.5927
Original Adult	**0.8476**	0.8461	0.8358	0.8289
+ 15% Noise	0.8032	**0.8219**	0.7630	0.7630
Original USPS	**0.9123**	0.9103	0.9103	0.7833
+ 15% Noise	0.8744	0.8739	**0.8789**	0.6846
Original OptDigits	0.9249	0.9355	**0.9416**	0.7958
+ 15% Noise	**0.8948**	**0.8948**	0.8770	0.6884

The testing set results using the value of ν with the best validation set accuracy are given in Table 5.2. LPBoost was very comparable with AdaBoost in terms of CPU time. As seen in Table 5.2, LPBoost is also comparable with AdaBoost in terms of classification accuracy when the validation set is used to pick the best parameter settings. All boosting methods outperform C4.5. In general, AdaBoost had the best performance with narrow margins. LPBoost and CRB performed comparable with AdaBoost.

The computational costs of 25 iterations of LPBoost (either variant) and AdaBoost were very similar. We provide some sample CPU times. These timings should be considered only rough estimates. Our experiments were performed on a cluster of IBM RS-6000s used in batch mode. Since the machines are not all identical and are subject to varying loads, run times vary considerable from run to run. For each dataset we give the seconds of CPU time on an RS-6000: Forest AdaBoost =717, LPBoost = 930; Adult AdaBoost = 107, LPBoost = 89; USPS AdaBoost = 208, LPBoost = 177; and Optdigits AdaBoost = 21, LPBoost = 24.

We also conducted experiments by boosting C4.5 on small datasets. Once again there was no strong evidence of superiority of any of the boosting approaches. In addition to six UCI datasets used in decision tree stumps experiments, we use four additional UCI datasets here. These are the House(16,435), Housing(13,506)[5], Pima(8,768), and Spam(57,4601) datasets. As in the decision tree stumps experi-

[5]The continuous response variable of Housing dataset was categorized at 21.5.

Table 5.3: Small Dataset Results from Boosting C4.5

Dataset	LPBoost	CRB	AdaBoost	C4.5
Cancer	0.9585 ± 0.0171	0.9628 ± 0.0245	0.9662 ± 0.0254	0.9447 ± 0.0248
Diagnostic	0.9649 ± 0.0263	0.9631 ± 0.0280	0.9705 ± 0.0186	0.9370 ± 0.0364
Heart	0.7913 ± 0.0624	0.7946 ± 0.0996	0.7867 ± 0.0614	0.7880 ± 0.0767
House	0.9586 ± 0.0339	0.9447 ± 0.0525	0.9511 ± 0.0417	0.9618 ± 0.0289
Housing	0.8538 ± 0.0476	0.8656 ± 0.0378	0.8785 ± 0.0393	0.8173 ± 0.0486
Ionosphere	0.9373 ± 0.0375	0.9259 ± 0.0604	0.9355 ± 0.0406	0.9158 ± 0.0520
Musk	0.8824 ± 0.0543	0.9055 ± 0.0490	0.9293 ± 0.0284	0.8344 ± 0.0340
Pima	0.7500 ± 0.0499	0.7279 ± 0.0483	0.7478 ± 0.0707	0.7286 ± 0.0455
Sonar	0.8173 ± 0.0827	0.8317 ± 0.0827	0.8140 ± 0.0928	0.7011 ± 0.0727
Spam	0.9557 ± 0.0086	0.9550 ± 0.0098	0.9518 ± 0.0092	0.9296 ± 0.0087

ments, we report results from 10-fold CV. Since the best ν value for LPBoost varies between 0.05 and 0.1 for the large datasets, we pick parameter $\nu = 0.07$ for the small datasets. Results are reported in Table 5.3. C4.5 performed the best on the House dataset. AdaBoost performed the best in four datasets out of ten. LPBoost and CRB had the best classification performance for three and two datasets respectively. When we drop CRB in Table 5.3, LPBoost would in this case perform the best in five datasets, although the parameter ν has not been tuned.

5.8 Discussion and Extensions

We have shown that LP formulations of boosting are both attractive theoretically in terms of generalization error bound and computationally via column generation. The LPBoost algorithm can be applied to any boosting problem formulated as an LP. We examined algorithms based on the 1-norm soft margin cost functions for support vector machines. A generalization error bound was found for the classification case. The LP optimality conditions allowed us to provide explanations for how the methods work. In classification, the dual variables act as misclassification costs. The optimal ensemble consists of a linear combination of weak learners that work best under the worst possible choice of misclassification costs. This explanation is closely related to that of [23]. For regression as discussed in the Barrier Boosting approach to a similar formulation [80], the dual multipliers act like error residuals to be used in a regularized least square problem. We demonstrated the ease of adaptation to other boosting problems by examining the confidence-rated and regression cases. Extensive computational experiments found that the method performed well versus AdaBoost both with respect to classification quality and solution time. Experimental results have shown that boosting of C4.5 decision trees improved the testing accuracy. From an optimization perspective, LPBoost has many benefits over gradient-based approaches: finite termination, numerical stability, well-defined convergence criteria, fast algorithms in practice, and fewer weak learners in the optimal ensemble. LPBoost may be more sensitive to inexactness of the base learning algorithm. But through modification of the base LP, we were able to obtain very good performance over a wide spectrum of datasets even in the boosting decision trees where the assumptions of the learning algorithm were violated. The questions of what is the best LP formulation for boosting and the best method for optimizing the LP remain open. Interior point column generation algorithms may be much more efficient. But clearly LP formulations for classification and regression are tractable using column generation, and should be the subject of further research.

CHAPTER 6
Conclusion

Turkish mustaches, or lack thereof, bristle with meaning.... Mustaches signal the difference between leftist (bushy) and rightist (drooping to the chin), between Sunni Muslim (clipped) and Alevi Muslim (curling to the mouth).

Wall Street Journal, May 15, 1997

Rules of thumb can help to enhance our lives. We probably learn these rules of thumb through the course of experience. Good rules could be regarded as "proverbs" or "rules to liveby". We might also have so called "stereotypes" about people or places. What makes someone a wise person is simply that how she/he can generalize well to solve the present difficulties faced using past experiences. What degrades a person's character is his/her willingness to accept stereotypes. It could be true that wise people have good rules in their lives to solve problems and to live in peace. In any case, we should listen to our conscience but not the stereotypes.

Stereotypes are analogous to poor generalization ability of the machine learning models. We proposed several innovative techniques in this research to improve the generalization ability of the machine learning models based on capacity control by using all the available information available. Our methods span a variety of learning methods: supervised, unsupervised and semi-supervised learning. As we indicated above, how well we deal with the new problems based on our experiences improves our lives. If we solve problems easily and in a proper way, it means that we have learned well from our experiences. Our aim was to develop measures and methods to provide the same confidence in machine learning techniques as well.

In Chapter 3, a novel method for semi-supervised learning that combines aspects of supervised and unsupervised learning techniques was introduced. The central focus was to take an unsupervised clustering method, label each cluster with the class membership, and simultaneously optimize the misclassification error of the resulting clusters. A linear combination of both cluster dispersion and cluster (class) impurity measures formed the objective function of learning process. The rationale behind this approach was that to avoid overfitting, the unsupervised component of the objective function acts as a form of regularization or capacity control during supervised learning.

The method allows the user to exploit any available unlabeled data during training since the cluster dispersion measure does not require class labels. Therefore, this approach can easily be adapted to transductive inference, the process of constructing a classifier using both the labeled training data and the unlabeled test data. We used two different measures for unsupervised information (cluster dispersion): Mean Square Error (MSE) and Davies-Bouldin Index (DBI). Experimental results show that using DBI for cluster dispersion instead of MSE improves transductive inference. Minimizing DBI results in compact and well separated clusters. DBI finds solutions using far fewer clusters than MSE with much greater accuracy.

There are two types of research contributions in Chapter 3. These contributions are in terms of both application of genetic algorithms and learning methods. Although floating-point genome representation had been around before, it was the first time such representation was used to solve clustering problems in genetic algorithms. We proposed a semi-supervised clustering method that used genetic algorithms. The parametric objective function allowed us to solve supervised, unsupervised and semi-supervised learning problems within the same model. The idea incorporating classification information into an unsupervised algorithm and using the resulting algorithm for transductive inference methods is applicable to many types of unsupervised learning. These are also promising areas of future research.

The semi-supervised clustering method discussed in Chapter 3 can also be generalized to regression problems. Since each cluster defines a neighborhood, we can predict the continuous variable within that neighborhood. Moreover, we can predict the continuous variable in the local neighborhoods. One could also implement scaling for variable selection in the semi-supervised framework. For each variable, a gene must be introduced to represent the scaling factor. Scaling can be either implemented within the same genome or can be used with another genome for scaling factors within the GA.

We examined mathematical models for semi-supervised support vector machines (S^3VM) in Chapter 4. We proposed a general S^3VM model that minimizes both the misclassification error and the function capacity based on all the available data. Three different functions for penalizing unlabeled points that fall in the margin were discussed. Our computational investigation focused on the minimum error formulation for the transductive inference problem. We converted this problem to a mixed-integer program that can be exactly solved using commercial integer programming packages. We proposed semi-supervised models both in the primal and dual spaces. We also implemented the primal model in the local neighborhoods. By using the MIP formulation with a local learning algorithm, a powerful scalable transductive inference method was created. Our computational experiments found that the local learning method was the most effective overall.

Since local methods performed well, further studies are needed to determine how best to select neighborhoods and to choose the parameters within the local S^3VM-MIP. In addition, efficient computational methods for defining locality are left for future work. Currently, for each point a local model is built. The result for the points other than query point are discarded. We might, as well, keep those results for future predictions. We also examined a noncovex quadratic optimization approach to S^3VM. Our computational studies were less conclusive using this approach. Quadratic transductive models are extremely slow. Better formulations are within the scope of the future work.

The methods in Chapters 3 and 4 were implemented in the feature space. In Chapter 5, we proposed a soft margin classifier (LPBoost) in the label space to solve the boosting problem. We have shown that LP formulations of boosting are attractive both theoretically (in terms of generalization error bound) and computationally (via column generation). The LPBoost algorithm can be applied to any boosting problem formulated as an LP. We examined algorithms based on the 1-norm soft margin cost functions for support vector machines. The LP optimality conditions allowed us to provide explanations for how the methods work. In classification, the dual variables act as misclassification costs. The optimal ensemble consists of a linear combination of weak learners that work best under the worst possible choice of misclassification costs.

We demonstrated the ease of adaptation to other boosting problems by examining the confidence-rated and regression cases. Extensive computational experiments found that the method performed well versus AdaBoost both with respect to classification quality and solution time. From an optimization perspective, LPBoost has many benefits over gradient-based approaches:

- Finite termination
- Numerical stability
- Well-defined convergence criteria
- Fast algorithms in practice
- Fewer weak learners in the optimal ensemble

LPBoost may be more sensitive to inexactness of the base learning algorithm. However, through modification of the base LP, we were able to obtain very good performance over a wide spectrum of datasets even in the boosting decision trees where the assumptions of the learning algorithm were violated. The questions of what is the best LP formulation for boosting and the best method for optimizing the LP remain open. Adapting efficient optimization methods for column generation algorithms such as interior point methods could be further investigated.

Research in support vector regression models is very active. Successful models such as the ϵ-Tuning model [92] exist to solve regression problems. The LP formulation of regression problems can also be solved by column generation techniques. In addition, for boosting regression as discussed in the Barrier Boosting approach to a similar formulation [80], the dual multipliers act like error residuals to be used in a regularized least square problem. However, clearly LP formulations for classification and regression are tractable using column generation, and should be the subject of further research.

LITERATURE CITED

[1] C. G. Atkeson, A. W. Moore, and S. Schaal. Locally weighted learning. *Artificial Intelligence Review*, 11:11–73, 1997.

[2] E. Bauer and R. Kohavi. An empirical comparison of voting classification algorithms: Bagging, boosting, and variants. *Machine Learning*, 36:105–139, 1999.

[3] K. P. Bennett. Global tree optimizaiton:a non-greedy decision tree algorithm. *Computing Science and Statistics*, 26:156–160, 1994.

[4] K. P. Bennett. Combining support vector and mathematical programming methods for classification. In B. Schölkopf, C. Burges, and A. Smola, editors, *Advances in Kernel Methods – Support Vector Machines*, pages 307–326, Cambridge, MA, 1999. MIT Press.

[5] K. P. Bennett. Kernel principal component analysis. In B. Schölkopf, C. Burges, and A. Smola, editors, *Advances in Kernel Methods – Support Vector Machines*, pages 327–352, Cambridge, MA, 1999. MIT Press.

[6] K. P. Bennett and J. A. Blue. Optimal decision trees. Technical Report R.P.I. Math Report No.214, Rennselaer Polytechnic Institute, Troy, NY, 1996.

[7] K. P. Bennett and J.A. Blue. A support vector machine approach to decision trees. Technical Report R.P.I. Math Report No. 97-100, Rensselaer Polytechnic Institute, Troy, NY, 1997.

[8] K. P. Bennett and E. J. Bredensteiner. Geometry in learning. In C. Gorini, E. Hart, W. Meyer, and T. Phillips, editors, *Geometry at Work*, pages 132–145, Washington, D.C., 2000. Mathematical Association of America. Available also at $http://www.rpi.edu/\sim bennek/geometry2.ps$.

[9] K. P. Bennett and A. Demiriz. Semi-supervised support vector machines. In D. Cohn M. Kearns, S. Solla, editor, *Advances in Neural Information Processing Systems 11*, pages 368–374, Cambridge, MA, 1999. MIT Press.

[10] K. P. Bennett, A. Demiriz, and John Shawe-Taylor. A column generation approach to boosting. In *Proceedings of International Conference on Machine Learning*, Stanford, California, 2000. To appear.

[11] K. P. Bennett and O. L. Mangasarian. Robust linear programming discrimination of two linearly inseparable sets. *Optimization Methods and Software*, 1:23–34, 1992.

[12] K. P. Bennett and O. L. Mangasarian. Bilinear separation in n-space. *Computational Optimization and Applications*, 4(4):207–227, 1993.

[13] A.M. Bensaid, L.O. Hall, J.C. Bezdek, and L.P. Clarke. Partially supervised clustering for image segmentation. *Pattern Recognition*, 29(5):859–871, 1996.

[14] D. P. Berstsekas. *Nonlinear Programming.* Aethena Scientific, Cambridge, MA, 1996.

[15] J.C. Bezdek. Classification of posture maintenance data with fuzzy clustering algorithms final report. Technical Report NASA CR-189940, NASA, 1992.

[16] J.C. Bezdek, T.R. Reichherzer, G.S. Lim, and Y. Attikiouzel. Multiple prototype classifier design. *IEEE Transactions on Systems, Man, and Cybernetics*, 28(1):67–79, 1998.

[17] J. Blue. *A hybrid of tabu search and local descent algorithms with applications in artificial intelligence.* PhD thesis, Rensselaer Polytechnic Institute, Troy, NY, 1998.

[18] A. Blum and T. Mitchell. Combining labeled and unlabeled data with co-training. In *Proceedings of the 1998 Conference on Computational Learning Theory*, Madison WI, 1998. ACM Inc.

[19] B. Boser, I. Guyon, and V. N. Vapnik. A training algorithm for optimal margin classifiers. In *Fifth Annual Workshop on Computational Learning Theory*, pages 144–152, Pittsburgh, 1992. ACM.

[20] L. Bottou. About direct regression and density estimation. Presentation at Learning Workshop, Snowbird, Utah, April 6-9, 1999.

[21] E. J. Bredensteiner and K. P. Bennett. Feature minimization within decision trees. *Computational Optimization and Applications*, 10:110–126, 1997.

[22] L. Breiman. Arcing the edge. Technical Report 486, Statistics Department, University of California at Berkeley, 1997.

[23] L. Breiman. Prediction games and arcing algorithms. *Neural Computation*, 11(7):1493–1517, 1999.

[24] L. Breiman, J. Friedman, R. Olshen, and C. Stone. *Classification and Regression Trees*. Wadsworth International, California, 1984.

[25] J. Buhmann. Stochastic algorithms for exploratory data analysis: Data clustering and data visualization. In M. Jordan, editor, *Learning in Graphical Models*. Kluwer Academics, 1997.

[26] C.J.C. Burges and B. Schölkopf. Improving the accuracy and speed of support vector machines. In M. Mozer, M. Jordan, and T. Petsche, editors, *Advances in Neural Information Processing Systems (NIPS) 9*, Cambridge, MA, 1997. MIT Press.

[27] V. Castelli and T. M. Cover. On the exponential value of labeled samples. *Pattern Recognition Letters*, 16:105–111, 1995.

[28] Z. Çataltepe and M. Magdon-Ismail. Incorporating test inputs into learning. In *Proceedings of the Advances in Neural Information Processing Systems, 10*, Cambridge, MA, 1997. MIT Press.

[29] P. Chaudhuri, W-D. Lu, and W-Y. Loh. Generalized regression trees. *Statistica Sinica*, 5:641–666, 1995.

[30] V. Cherkassky and F. Mulier. *Learning from Data.* Wiley Inter-Science, 1998.

[31] V. Cherkassy, Y. Kim, and F Mulier. Constrained topological maps for regression and classification. In *Proceedings of Int. Conf. on Neural Information Processing, New Zealand*, November 1997.

[32] C. Cortes and V. N. Vapnik. Support vector networks. *Machine Learning*, 20:273–297, 1995.

[33] CPLEX Optimization Incorporated, Incline Village, Nevada. *Using the CPLEX Callable Library*, 1994.

[34] N. Cristianini and J. Shawe-Taylor. *An Introduction to Support Vector Machines.* Cambridge University Press, 2000.

[35] R. Cucchiara. Genetic algorithms for clustering in machine vision. *Machine Vision and Applications*, 11:1–6, 1998.

[36] D.L. Davies and D.W. Bouldin. A cluster separation measure. *IEEE Transcations on Pattern Analysis and Machine Intelligence*, 1(2):224–227, 1979.

[37] A. Demiriz and K. P. Bennett. Optimization approaches to semi-supervised learning. In M. C. Ferris, O. L. Mangasarian, and J.-S. Pang, editors, *Applications and Algorithms of Complementarity*, Boston, 2000. Kluwer Academic Publishers. To Appear.

[38] A. Demiriz, K. P. Bennett, and M. J. Embrechts. A genetic algorithm approach for semi-supervised clustering. Manuscript submitted for publication $http://www.rpi.edu/\sim demira/jsmc.ps$, 1999.

[39] A. Demiriz, K. P. Bennett, and M. J. Embrechts. Semi-supervised clustering using genetic algorithms. In C. Dagli, editor, *Proceedings of ANNIE99 (Artificial Neural Networks in Engineering).* ASME Press, 1999.

[40] A. Demiriz, K. P. Bennett, and John Shawe-Taylor. Linear programming boosting via column generation. Manuscript submitted for publication $http://www.rpi.edu/\sim demira/mlj.ps$, 2000.

[41] L. Devroye, L. Györfi, and G. Lugosi. *A probabilistic theory of pattern recognition.* Springer, New York, NY, 1996.

[42] E. Diday. From data to knowledge:probabilistic objects for a symbolic data analysis. In I. Cox, P. Hansen, and B. Julesz, editors, *Partitioning Data Sets*, pages 35–53. American Mathematical Society, 1995.

[43] H. Drucker, C.J.C. Burges, L.Kaufman, A. Smola, and V. Vapnik. Support vector regression machines. In M. Mozer, M. Jordan, and T. Petsche, editors, *Advances in Neural Information Processing Systems (NIPS) 9*, Cambridge, MA, 1997. MIT Press.

[44] M. J. Embrechts, R. Kewley, and C. Breneman. Computationally intelligent data mining for the automated design and discovery of novel pharmaceuticals. In C. Dagli, editor, *Intelligent Engineering Systems throught Artificial Neural Networks.* ASME Press, 1998.

[45] B.S. Everitt. *Cluster Analysis.* John Wiley & Sons Inc., 1974.

[46] D. Fisher. Knowledge acquisition via incremental conceptual clustering. *Machine Learning*, 2, 1987.

[47] R. Fourer, D. Gay, and B. Kernighan. *AMPL A Modeling Language for Mathematical Programming.* Boyd and Frazer, Danvers, MA, 1993.

[48] J.H. Friedman. Flexible metric nearest neighbor classification, 1994. $http://www-stat.stanford.edu/\sim jhf/$.

[49] J.H. Friedman, R. Kohavi, and Y. Yun. Lazy decision trees. In *Proceedings of AAAI-96*, Menlo Park, CA, 1996. AAAI-Press.

[50] D. E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning.* Addison–Wesley, Reading, MA, 1989.

[51] A. J. Grove and D. Schuurmans. Boosting in the limit: Maximizing the margin of learned ensembles. In *Proceedings of Fifteenth National Conference on Artificial Intelligence (AAAI-98)*. AAAI Press, 1998.

[52] I. Guyon and D. G. Stork. Linear discriminant and support vector classifiers. In A.J. Smola, P.L. Bartlett, B. Schölkopf, and D. Schuurmans, editors, *Advances in Large Margin Classifiers*, pages 147–169, Cambridge, MA, 2000. MIT Press.

[53] P. Hansen and B. Jaumard. Cluster analysis and mathematical programming. *Mathematical Programming*, 79:191–215, 1997.

[54] T. Hastie and R. Tibshirani. Discriminant adaptive nearest neighbor classification. *IEEE PAMI*, 18:607–616, 1996.

[55] T. Hofmann and J. Buhmann. Hierarchical pairwise data clustering by mean-field annealing. In *Proceedings of the International Conference on Artificial Neural Networks (ICANN'95)*, pages 197–202, 1995.

[56] T. Hofmann, J. Puzicha, and J. M. Buhmann. Deterministic annealing for unsupervised texture segmentation. In *Proceedings of the International Workshop on Energy Minimization Methods in Computer Vision and Pattern Recognition (EMMCVPR'97)*, Venice, 1997.

[57] F. Höppner, F. Klawon, R. Kruse, and T. Runkler. *Fuzzy cluster analysis.* Wiley, 1999.

[58] A.K. Jain and R.C. Dubes. *Algorithms for Clustering Data.* Prentice Hall, Englewood Cliffs, N.J., 1988.

[59] T. Joachims. Text categorization with support vector machines: Learning with many relevant features. In *European Conference on Machine Learning(ECML)*, 1998.

[60] T. Joachims. Transductive inference for text classification using support vector machines. In *International Conference on Machine Learning*, 1999.

[61] T. Kohonen. Learning vector quantitization. *Neural Networks*, 1, 1988.

[62] L.I. Kuncheva and J.C. Bezdek. Nearest prototype classification: Clustering, genetic algorithms, or random search? *IEEE Transactions on Systems, Man, and Cybernetics*, 28(1):160–164, 1998.

[63] S. Lawrence, A. C. Tsoi, and A. D. Back. Function approximation with neural networks and local methods: Bias, variance and smoothness. In Peter Bartlett, Anthony Burkitt, and Robert Williamson, editors, *Australian Conference on Neural Networks, ACNN 96*, pages 16–21. Australian National University, 1996.

[64] O. L. Mangasarian. Arbitrary norm separating plane. *Operations Research Letters*, 24(1-2), 1999.

[65] O. L. Mangasarian. Generalized support vector machines. In A. Smola, P. Bartlett, B. Schölkopf, and D. Schuurmans, editors, *Advances in Large Margin Classifiers*, pages 135–146, Cambridge, MA, 2000. MIT Press. ftp://ftp.cs.wisc.edu/math-prog/tech-reports/98-14.ps.

[66] L. Mason, J. Baxter, P. Bartlett, and M. Frean. Boosting algorithms as gradient descent in function space, 1999.

[67] A. McCallum and K. Nigam. Employing em and pool-based active learning for text classification. In *Proceedings of the 15th International Conference on Machine Learning (ICML-98)*, pages 350–358, 1998.

[68] P.M. Murphy and D.W. Aha. *UCI repository of machine learning databases.* Department of Information and Computer Science, University of California, Irvine, California, 1992.

[69] F. Murtagh. *Multidimensional Clustering Algorithms.* Physica-Verlag, Vienna, 1985.

[70] C.A. Murthy and N. Chowdhury. In search of optimal clusters using genetic algorithms. *Pattern Recognition Letters*, 17:825–832, 1996.

[71] D. R. Musser and A. Saini. *STL Tutorial and Reference Guide: C++ Programming with the Standard Template Library.* Addison-Wesley, 1996.

[72] S.G. Nash and A. Sofer. *Linear and Nonlinear Programming.* McGraw-Hill, New York, NY, 1996.

[73] R. T. Ng and J. Han. Efficient and effective clustering for spatial data mining. In *Proceedings of 20th VLDB Conference Santiago, Chile*, 1994.

[74] K. Nigam, A. McCallum, S. Thrun, and T. Mitchell. Learning to classify text from labeled and unlabeled documents. In *Proceedings of the 15th National Conference on Artificial Intelligence (AAAI-98)*, 1998.

[75] S. Odewahn, E. Stockwell, R. Pennington, R Humphreys, and W Zumach. Automated star/galaxy discrimination with neural networks. *Astronomical Journal*, 103(1):318–331, 1992.

[76] W. Pedrycz. Algorithms of fuzzy clustering with partial supervision. *Pattern Recognition Letters*, 3:13–20, 1985.

[77] J. R. Quinlan. Induction of decision trees. *Machine Learning*, 1:81–106, 1984.

[78] J. R. Quinlan. *C4.5: Programs for Machine Learning.* Morgan Kaufmann, 1993.

[79] J.R. Quinlan. Bagging, boosting, and C4.5. In *Proceedings of the 13th National Conference on Artificial Intelligence*, Menlo Park, CA, 1996. AAAI Press.

[80] G. Rätsch, S. Mika, T. Onoda, S. Lemm, and K.-R. Müller. Barrier boosting. Technical report, 2000. Private communication - submitted for publication.

[81] G. Rätsch, B. Schölkopf, A.J. Smola, S. Mika, T. Onoda, and K-R. Müller. Robust ensemble learning. In A.J. Smola, P. L. Bartlett, B. Schölkopf, and D. Schuurmans, editors, *Advances in Large Margin Classifiers*, pages 208–222, Cambridge, MA, 2000. MIT Press.

[82] G. Rätsch, B. Schölkopf, A.J. Smola, K-R. Müller, T. Onoda, and S. Mika. ν-arc ensemble learning in the presence of outliers. In S. Solla, T.K. Leen, and K-R. Müller, editors, *Advances in Neural Information Processing Systems 12*, Cambridge, MA, 2000. MIT Press.

[83] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning internal representations by error propagation. In *Parallel distributed processing:Explorations in the macrostructures of cognition*, volume 1, pages 318–362, Cambridge, MA, 1986. Bradford Books.

[84] A. Rusch and R. Wille. Knowledge spaces and formal concept analysis. In H. Boch and W. Polasek, editors, *Data Analysis and Information Systems*, pages 427–436. Springer, 1996.

[85] M. Sarkar, B. Yegnanarayana, and D. Khemani. Clustering algorithm using an evolutionary programming-based approach. *Pattern Recognition Letters*, 18:975–986, 1997.

[86] SAS Institute Inc., Cary, North Carolina. *SAS Language and Procedures:Usage, Version 6*, 1989.

[87] R. Schapire, Y. Freund, P. Bartlett, and W. Sun Lee. Boosting the margin: A new explanation for the effectiveness of voting methods. *Annals of Statistics*, 26(5):1651–1686, 1998.

[88] R. Schapire and Y. Singer. Improved boosting algorithms using confidence-rated predictions. In *Proceedings of the Eleventh Conference on Computational Learning Theory, COLT'98*, pages 80–91, 1998. to appear in Machine Learning.

[89] B. Schölkopf, A. Smola, and K-R. Müller. Kernel principal component analysis. In *Proceedings of ICANN'97*, pages 583–588. Springer Lecture Notes in Computer Science, 1997.

[90] B. Schölkopf, A. Smola, and K-R. Müller. Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation*, 10(5):1299–1319, 1998.

[91] B. Schölkopf, K. Sung, and V. Vapnik. Comparing support vector machines with gaussian kernel to radial basis function classifiers. Technical Report A.I. Memo No 1599, MIT, 1996.

[92] A. Smola, B. Schölkopf, and G. Rätsch. Linear programs for automatic accuracy control in regression. In *Proceedings ICANN'99, Int. Conf. on Artificial Neural Networks*, Berlin, 1999. Springer.

[93] M.O. Stitson, J.A.E. Weston, and V. Vapnik. Theory of support vector machines. Technical Report CSD-TR-96-17, Royal Holloway, University of London, 1996.

[94] K. Tsuda. Optimal hyperplane classifier with adaptive norm. Technical Report ETL TR-99-9, Electrotechnical Laboratory, Japan, 1999.

[95] M. Vaidyanathan, R.P. Velthuizen, P. Venugopal, L.P. Clarke, and L.O. Hall. Tumor volume measurements using supervised and semi-supervised mri segmentation. In C. H. Dagli, editor, *Proceedings of Artificial Neural Networks in Engineering (ANNIE)*, 1994.

[96] V. Vapnik. The support vector method of function estimation. In C. M. Bishop, editor, *Neural Networks and Machine Learning*, volume 168, pages 239–268. Springer, 1998. NATO ASI Series.

[97] V. N. Vapnik. *Estimation of dependencies based on empirical Data.* Springer, New York, 1982. English translation, Russian version 1979.

[98] V. N. Vapnik. *The Nature of Statistical Learning Theory.* Springer Verlag, New York, 1995.

[99] V. N. Vapnik. *Statistical Learning Theory.* Wiley Inter-Science, 1998.

[100] V. N. Vapnik and A. Ja. Chervonenkis. *Theory of Pattern Recognition.* Nauka, Moscow, 1974. In Russian.

[101] V. N. Vapnik, S. Golowich, and A. Smola. Support vector method for function approximation, regression estimation, and signal processing. In

M. Mozer, M. Jordan, and T. Petsche, editors, *Advances in Neural Information Processing Systems (NIPS) 9*, Cambridge, MA, 1997. MIT Press.

[102] M. Wall. *GAlib: A C++ Library of Genetic Algorithm Components.* MIT, http://lancet.mit.edu/ga/, 1996.