Convex Large Margin Training Techniques

Unsupervised, Semi-supervised, and Robust

Support Vector Machines

by

Linli Xu

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Doctor of Philosophy
in
Computer Science

Waterloo, Ontario, Canada, 2007

Abstract

Support vector machines (SVMs) have been a dominant machine learning technique for more than a decade. The intuitive principle behind SVM training is to find the maximum margin separating hyperplane for a given set of binary labeled training data. Previously, SVMs have been primarily applied to *supervised* learning problems, where target class labels are provided with the data. Developing *unsupervised* extensions to SVMs, where no class labels are given, turns out to be a challenging problem. In this dissertation, I propose a principled approach for unsupervised and semi-supervised SVM training by formulating convex relaxations of the natural training criterion: find a (constrained) labeling that would yield an optimal SVM classifier on the resulting labeled training data. This relaxation yields a semidefinite program (SDP) that can be solved in polynomial time. The resulting training procedures can be applied to two-class and multi-class problems, and ultimately to the multivariate case, achieving high quality results in each case. In addition to unsupervised training, I also consider the problem of reducing the outlier sensitivity of standard supervised SVM training. Here I show that a similar convex relaxation can be applied to improve the robustness of SVMs by explicitly suppressing outliers in the training process. The proposed approach can achieve superior results to standard SVMs in the presence of outliers.

Acknowledgements

This thesis concludes several years of work and research in my life, during this time I have been so fortunate to have the support and encouragement from many people. Here I would like to take this opportunity to express my sincere appreciation to each of them.

I am deeply grateful to my supervisor, Professor Dale Schuurmans. Throughout my research, Dale has been offering tremendous and insightful guidance and help. I have benefitted significantly from his broad knowledge and active research attitude, and I really appreciate how supportive and considerate Dale has been throughout this process.

I would like to express my appreciation to my committee members, Shai Ben-David, Pascal Poupart, Daniel Brown, Romy Shioda and Thorsten Joachims for their time and effort.

I am also thankful to my collaborators, Koby Crammer, Dana Wilkinson, Finnegan Southey, James Neufeld and Bryce Larson for their help in my research.

Moreover, my appreciation goes to my friends and colleagues for their generous support and great companionship. I was not lonely during the journey thanks to them.

Finally, I want to thank my parents, my brother and his wife, together with my little niece Yuchen for their absolute and constant support. No matter how far away I am from them, they are always there for me. It is my great fortune in life to have them as my source of strength, happiness and love.

To my parents

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Supervised, semi-supervised and unsupervised learning are three of the dominant topics of machine learning research. In supervised learning, a label is provided with each data example given to the training procedure, whereas in unsupervised learning, no labels are provided. Semi-supervised learning is an intermediate form of the problem where only a subset of the target labels are provided during training. Although there have been a great number of principles proposed for supervised, semi-supervised and unsupervised training techniques respectively, most principles proposed in the literature are not applicable to all three problems, but are instead only focused on one or two of them. What most proposals fail to provide is a unifying perspective on all three of these different types of machine learning problems. One of my motivations in this thesis is to investigate whether unifying machine learning principles exist that can be used to combine different types of these learning problems in a common framework. The specific methodology I follow is to exploit one particular form of supervised learning principle and modify it to address unsupervised learning, with the goal of achieving a simple and unified approach to solving various types of learning problems, including supervised, unsupervised and semi-supervised learning.

The specific supervised learning principle I investigate in this thesis is the large margin principle of support vector machines (SVMs). Support vector machines introduced the notion of large margin training criteria for classification learning and have been one of the most commonly applied machine learning methods in the past decade (Cortes and Vapnik, 1995; Boser et al., 1992). The simplest version of the idea is to find a hyperplane that correctly separates binary labeled training data with the maximum possible margin, and thus to reduce the risks of future misclassifications. This approach has the advantage that training can be performed efficiently using quadratic programming. Moreover, kernel techniques and feature maps can be introduced to achieve nonlinear decision boundaries without affecting efficiency.

To adapt support vector machines to *unsupervised* learning problems, the training objective needs to be modified to finding a *labeling* that results in a large margin classifier. However, developing unsupervised extensions to SVMs turns out to be a challenging problem, since naive approaches require one to check an exponential number of possible label configurations. The apparent hardness of this problem has motivated researchers to heuristic (Joachims, 1999) or branch and bound (Vapnik, 1995) approaches in previous research.

Meanwhile, new opportunities for developing effective machine learning techniques are offered by the field of semidefinite programming (SDP) (Vandenberghe and Boyd, 1996), which extends the range of practical optimization methods beyond the current unconstrained, linear and quadratic programming techniques common in machine learning research. Recent progress in the field of semidefninte programming has provided viable techniques that have reasonable efficiency characteristics (Boyd and Vandenberghe, 2004), including polynomial run-time guarantees (Nesterov and Nemirovskii, 1994). In fact, semidefinite programming has already started to prove its utility in machine learning research. Lanckriet et al. (2004) show how semidefnite programming can be used to optimize the kernel matrix for a supervised SVM. Weinberger et al. (2004) investigate how to learn a kernel matrix for nonlinear dimensionality reduction based on semidefinite programming. Similar techniques are used in the correlation clustering approach of (Swamy, 2004). Even though the number of such results remain modest, the initial achievements are impressive.

The above ideas lead to the first part of my thesis work—a principled approach to unsupervised SVM training by formulating *semidefinite* relaxations of the natural training criterion. That is, rather than formulate heuristic algorithms, as done in past research (Joachims, 1999), I instead formulate principled convex relaxations that can be solved in polynomial time. Moreover, I extend previous research by considering a fully unsupervised case (no training labels), before extending the training algorithms naturally to the semi-supervised case. Importantly, this approach not only can be applied to the binary class problem (Xu et al., 2004), but also can be further extended to the multi-class case (Xu and Schuurmans, 2005), making it applicable to a wider range of natural problems.

Nevertheless, both the binary and multi-class algorithms focus on solving problems in the simplest univariate model, where the system only learns to predict a single label. In many applications such as speech recognition (Jelinek, 1998), natural language processing (Jurafsky and Martin, 2000) and biological sequence analysis (Durbin et al., 1998), one would hope to train a model with a *sequence* of labels; hence the learning task involves learning to make *structured* predictions. This is an extension of the standard supervised learning to the multivariate setting with structured, non-scalar predictions $\mathbf{y}$. The challenge here is that for each component $\hat{y}_i$, one should not only consider the input $\mathbf{x}$, but also the direct correlations of its neighboring components. Recently there have been some significant advances in learning structured predictors, including conditional random fields (Lafferty et al., 2001), discriminative sequence training (Tsochantaridis et al., 2004; Altun et al., 2003) and maximum margin Markov networks (Taskar et al., 2003). However, this work primarily focuses on the *supervised* case, where the label sequences are provided during training. Supervision is a serious limitation in practice because labeled data is usually very expensive and hard to obtain in many applications—particularly in the multivariate case—whereas unlabeled data is cheap and abundant.

For *unsupervised* learning of structured models, such as hidden Markov models, most previous research has focused on using the EM algorithm, whose objective is to maximize the likelihood of the observed inputs $p(\mathbf{x})$. However, in supervised training it has been observed that discriminative training—that is, maximizing the conditional likelihood of the outputs given the inputs $p(\mathbf{y}|\mathbf{x})$—is usually more advantageous because the conditional model works as the structured predictor of a joint labeling $\mathbf{y}$ for an input sequence $\mathbf{x}$. Another shortcoming of the objective used by EM is that it is not concave, which means that EM fails to guarantee a global solution to the problem. To address these problems, I extend my results on unsupervised discriminative (large margin) training to the structured prediction case normally tackled by EM. In particular, I consider maximum margin Markov networks (Taskar et al., 2003), which are directly a multivariate version of SVMs, and develop a convex algorithm for *unsupervised* maximum margin Markov networks (Xu et al., 2006b), using the techniques proposed for univariate SVMs.

A final problem I address in this thesis is one of the well known risks of large margin training methods: sensitivity to outliers. In this part of my work the learning problem is supervised; that is the labels of the data are given. Although, intuitively, the notion of large margin reduces the risk of future misclassifications by separating the classes as widely as possible, the simplest version of maximum margin principle produces poor results on non-separable data. The problem is that the solution hyperplane becomes determined by the misclassified data furthest from the decision hyperplane in this case, which results in a breakdown in the performance, both in theory and practice. These risks are normally mitigated by using a soft margin criterion to reduce outlier sensitivity. However, outlier points still tend to have a maximal influence in determining the decision hyperplane, since they normally have the largest hinge loss. To mitigate this problem, I propose a new approach to suppress outliers explicitly in SVM training. I do so by augmenting each example with an auxiliary variable indicating whether the data point is an outlier or not. To optimize the objective over the indicator variables and SVM parameters jointly, I develop semidefinite relaxations that yield a practical convex optimization formulation of the training problem (Xu et al., 2006a).

1.1 Thesis Contributions

In this thesis I present four main contributions:

- **Unsupervised and semi-supervised two-class support vector machines**
  First, I propose a new method for unsupervised learning based on finding maximum margin hyperplanes through data. By reformulating the problem in terms of the implied equivalence relation matrix, the problem can be posed as a convex integer program. Although this still yields a difficult computational problem, the hard-clustering constraints can be relaxed to a soft-clustering formulation which can be feasibly solved with a semidefinite program. Since this clustering technique only depends on the data through the kernel matrix, nonlinear clustering can easily be

achieved in the same manner as spectral clustering. Another benefit of this approach is that it leads naturally to a semi-supervised training method for support vector machines. By maximizing the margin simultaneously on labeled and unlabeled training data, state of the art performance can be achieved by using a single, integrated learning principle.

- **Unsupervised and semi-supervised multi-class support vector machines**
  I then extend the previous two-class formulation to the case where more than two classes exist. This is based on the multi-class SVM algorithm proposed in (Crammer and Singer, 2001). The resulting unsupervised algorithm can also be easily augmented to semi-supervised problems. With this extension from binary to multi-class problems, one is now able to deal with a wider range of natural data sets. Moreover, the further extension to the more complex structured predictors is made possible.

- **Unsupervised large margin learning of structured predictors**
  I then propose a new unsupervised algorithm for training structured predictors. This new approach has the nice properties of being discriminative, convex, and it avoids the use of EM. The idea is to formulate an unsupervised version of structured learning methods, such as maximum margin Markov networks, that can be trained via semidefinite programming. The result is a discriminative training criterion for structured predictors (like hidden Markov models) that remains unsupervised and does not create local minima. To reduce training cost, the training procedure is reformulated to mitigate the dependence on semidefinite programming, and finally a heuristic procedure that avoids semidefinite programming entirely is proposed.

- **Robust support vector machine training via convex outlier ablation**
  Finally based on the techniques developed above, I propose a new approach to robust supervised SVM training that explicitly incorporates outlier suppression in the training process. In particular, I show that outlier detection can be encoded in the large margin training principle of support vector machines. By expressing a convex relaxation of the joint training problem as a semidefinite program, one can use this approach to robustly train a support vector machine while suppressing outliers.

1.2 Publication Notes

The work presented in this thesis has been published in several refereed conference proceedings. The work in Chapter 3, 4 and 5 on unsupervised two-class, multi-class and structured-output SVMs were published in in (Xu et al., 2004), (Xu and Schuurmans, 2005) and (Xu et al., 2006b) respectively, and the work on robust SVM training was published in (Xu et al., 2006a).

Chapter 2

Background

Before presenting the main contributions of my thesis research, I first review the relevant background. My thesis contributions combine ideas from many different sub-areas of machine learning and optimization research. Hence, I review four distinct topics that comprise the main sources of ideas I will use below: support vector machines, unsupervised classification and clustering, hidden Markov models, and semidefinite programming.

First, to establish the foundations of the whole thesis, the underlying concept of large margin based training, and in particular *supervised* training algorithms for SVMs are first introduced. Then to provide a sufficient background on unsupervised learning (clustering), I present a brief review of unsupervised learning approaches that deal with single-variable models. Subsequently, I introduce structured prediction models, and in particular discuss supervised and unsupervised training algorithms for hidden Markov models that I will extend in my own work in Chapter 5. Finally, I introduce some basic concepts of semidefinite programming that will be exploited in the algorithmic approaches I subsequently pursue throughout the thesis.

2.1 Support Vector Machines

Supervised classification learning is one of the most fundamental tasks in machine learning. Essentially, the problem is to observe a finite number of labeled data objects from two (or more) classes, and then learn a classifier that can accurately assign new data objects to the appropriate class. There is a variety of methods in literature for classification learning. Among these, support vector machines (SVMs) (Cortes and Vapnik, 1995; Boser et al., 1992) have been the most dominant technique over the past decade because of their effective performance and elegant theoretical properties.

In this section, I will first introduce the basic two-class SVM, and after that discuss extensions to multi-class SVMs.

2.1.1 Two-class Support Vector Machines

Assume we are given a set of labeled training examples $(\mathbf{x}^1, y^1), ..., (\mathbf{x}^n, y^n)$, where each example is assigned to one of the two classes $y^i \in \{-1, +1\}$. If one needs to learn a classifier from these labeled data, a natural thing to do is to find a linear discriminant $f_{\mathbf{w},b} = \mathbf{w} \cdot \mathbf{x} + b$ to separate the negative examples from the positive ones (for now we assume the data can be linearly separated). Later a new data point $\mathbf{x}_{new}$ can be classified according to this linear discriminant by

$$\hat{y}_{new} = \mathrm{sgn}(\mathbf{w} \cdot \mathbf{x}_{new} + b) \tag{2.1}$$

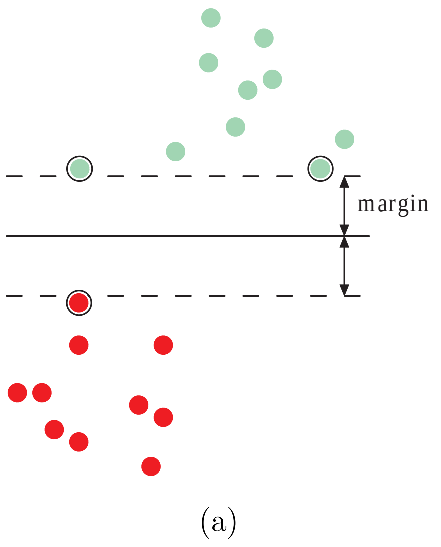
A problem with this method is that there generally exists more than one linear discriminant that is able to separate the data, so a criterion is needed to help decide which linear discriminant is better than the others.

A simple but effective way to solve this problem is to choose a linear discriminant that separates the data correctly and is maximally distant from the nearest data points; see Figure 2.1(a). Specifically, if we have a linear discriminant $f_{\mathbf{w},b} = \mathbf{w} \cdot \mathbf{x} + b$, then the distance of a data point $(\mathbf{x}^i, y^i)$ to the linear discriminant can be computed by

$$dist_i = \frac{|\mathbf{w} \cdot \mathbf{x}^i + b|}{\|\mathbf{w}\|} \tag{2.2}$$

Furthermore, if we add a sign to $dist_i$ to indicate whether the linear discriminant has classified the ith data point correctly, the signed distance can be computed as

$$\begin{aligned} sdist_i &= y^i \cdot \mathrm{sgn}(\mathbf{w} \cdot \mathbf{x}^i + b) \cdot dist_i \\ &= y^i \cdot \mathrm{sgn}(\mathbf{w} \cdot \mathbf{x}^i + b) \frac{|\mathbf{w} \cdot \mathbf{x}^i + b|}{\|\mathbf{w}\|} \end{aligned}$$



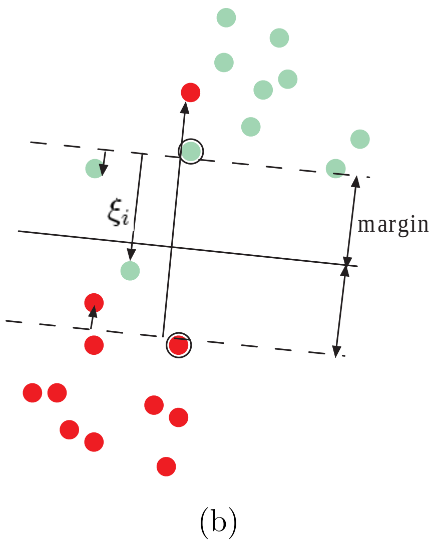


Figure 2.1: Support vector classifiers. The solid line is the decision boundary while the dashed lines show the margins. Circled data points are the support vectors. (a) Depicts a hard margin SVM, and (b) depicts a soft margin SVM. The slack value ξ_i for a point indicates how distant it is on the wrong side of the margin.

$$= \frac{y^i(\mathbf{w} \cdot \mathbf{x}^i + b)}{\|\mathbf{w}\|} \tag{2.3}$$

where we can see that if a data point is classified correctly, the signed distance of this point to the linear discriminant will be positive, and negative otherwise.

In this way, the minimum signed distance of all the data points to the linear discriminant is

$$\min_i \frac{y^i(\mathbf{w} \cdot \mathbf{x}^i + b)}{\|\mathbf{w}\|} \tag{2.4}$$

If we define (2.4) as the *margin* of the linear discriminant, then our goal is to find a linear discriminant with *maximum margin*. Thus, the problem can be formulated as

$$\gamma^* = \max_{\mathbf{w},b} \min_{i=1}^{n} \frac{y^i(\mathbf{w} \cdot \mathbf{x}^i + b)}{\|\mathbf{w}\|} \tag{2.5}$$

where the optimal value of this optimization problem is the maximum margin γ^* .

To turn (2.5) into a tractable optimization problem, we require a few steps of derivation:

$$\begin{aligned}
\gamma^* &= \max_{\mathbf{w},b} \min_{i=1}^{n} \frac{y^i(\mathbf{w}\cdot\mathbf{x}^i+b)}{\|\mathbf{w}\|} \\
&= \max_{\mathbf{w},b,\gamma} \gamma \quad \text{subject to } \frac{y^i(\mathbf{w}\cdot\mathbf{x}^i+b)}{\|\mathbf{w}\|} \geq \gamma \;\; \forall i \\
&= \max_{\mathbf{w},b,\gamma} \gamma \quad \text{subject to } y^i(\mathbf{w}\cdot\mathbf{x}^i+b) \geq \gamma\|\mathbf{w}\| \;\; \forall i \\
&= \max_{\mathbf{w},b} \frac{1}{\|\mathbf{w}\|} \quad \text{subject to } y^i(\mathbf{w}\cdot\mathbf{x}^i+b) \geq 1 \;\; \forall i \\
\gamma^{*-2} &= \min_{\mathbf{w},b} \|\mathbf{w}\|^2 \quad \text{subject to } y^i(\mathbf{w}\cdot\mathbf{x}^i+b) \geq 1 \;\; \forall i \qquad (2.6)
\end{aligned}$$

where γ^{*-2} is the inverse square of the optimal margin.

The optimization problem (2.6) is a convex quadratic program[1] (Boyd and Vandenberghe, 2004), and can be solved efficiently by various techniques (J.C.Platt, 1998). The resulting optimal classifier is called a Support Vector Machine (SVM).

Soft margin support vector machines

The algorithm above is called a *hard margin SVM* because of the assumption that the data is linearly separable. This assumption is enforced by the linear constraints $y^i(\mathbf{w}\cdot\mathbf{x}^i+b) \geq 1$ in (2.6), which are equivalent to $[1 - y^i(\mathbf{w}\cdot\mathbf{x}^i+b)]_+ = 0$.[2] However, linear separability is generally not the case in real applications, hence, a *soft margin SVM* is introduced to cope with noisy data.

In a soft margin SVM, we cope with non-separability by allowing some of the linear constraints $[1 - y^i(\mathbf{w}\cdot\mathbf{x}^i+b)]_+ = 0$ to be unsatisfied. Meanwhile, a penalty is needed for violating the constraints. The resulting optimization problem is

$$\gamma^{*-2} = \min_{\mathbf{w}} \frac{\beta}{2}\|\mathbf{w}\|^2 + \sum_{i=1}^{n}[1 - y^i(\mathbf{w}\cdot\mathbf{x}^i+b)]_+ \qquad (2.7)$$

where the penalty term $[1 - y^i(\mathbf{w}\cdot\mathbf{x}^i+b)]_+$ is called *hinge loss*, and β is the regularization parameter. From Figure 2.2 we can see that hinge loss is an upper bound on the misclassification error. The augmented soft margin SVM is now equivalent to minimizing regularized

[1]A quadratic program is an optimization problem with quadratic objective and linear constraints.
[2]The notation $[x]_+$ denotes the positive component of the argument, i.e. $[x]_+ = \max(x, 0)$.

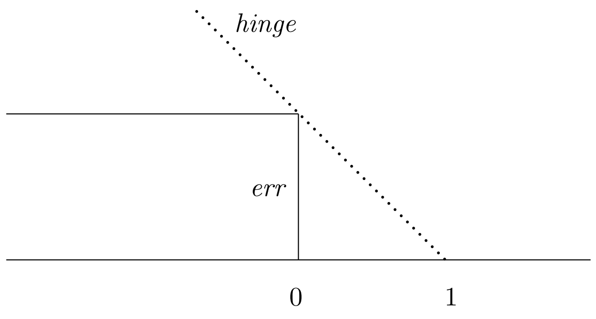


Figure 2.2: Margin losses as a function of $y(\mathbf{w}\cdot\mathbf{x}+b)$. The solid line depicts misclassification error, *err*, and the dotted line depicts hinge loss, *hinge*. Note that *hinge* $\geq$ *err*.

hinge loss, and it can be reformulated by adding a "slack" variable ξ_i to each example, or equivalently to each constraint in (2.6). The new optimization problem, equivalent to (2.7), is

$$\gamma^{*-2} = \min_{\mathbf{w},b,\boldsymbol{\xi}} \frac{\beta}{2}\|\mathbf{w}\|^2 + \sum_{i=1}^{n}\xi_i \quad \text{subject to} \quad \begin{array}{l} y^i(\mathbf{w}\cdot\mathbf{x}^i + b) \geq 1 - \xi_i \;\; \forall i \\ \xi_i \geq 0 \;\; \forall i \end{array} \tag{2.8}$$

Where $\xi_i > 0$, $\mathbf{x}^i$ is allowed to lie on the wrong side of margin (Figure 2.1(b)), and the degree of slackness is controlled by the β parameter. Thus, β is also known as the slack parameter.

Dual support vector machines

The quadratic program in (2.8) is called the primal support vector machine. One can obtain the dual to this quadratic program by introducing a dual variable for each constraint and formulating the Lagragian function (Boyd and Vandenberghe, 2004):

$$L = \frac{\beta}{2}\|\mathbf{w}\|^2 + \sum_{i=1}^{n}\xi_i - \sum_{i=1}^{n}\lambda_i[y^i(\mathbf{x}^i\cdot\mathbf{w}+b) - (1-\xi_i)] - \sum_{i=1}^{n}\mu_i\xi_i \quad \boldsymbol{\lambda},\boldsymbol{\mu} \geq 0 \tag{2.9}$$

By minimizing over $\mathbf{w}$, b, and $\boldsymbol{\xi}$, setting the respective derivatives to zero, we can obtain

$$\mathbf{w} = \frac{1}{\beta}\sum_{i=1}^{n}\lambda_i y^i \mathbf{x}^i \tag{2.10}$$

$$\sum_{i=1}^{n} \lambda_i y^i = 0 \tag{2.11}$$

$$\boldsymbol{\lambda} = \mathbf{e} - \boldsymbol{\mu} \tag{2.12}$$

together with all the non-negative constraints $\boldsymbol{\lambda}, \boldsymbol{\mu}, \boldsymbol{\xi} \geq 0$, where $\mathbf{e}$ is a vector of all ones. By substituting (2.10)-(2.12) into (2.9) and adding the constraints on dual variables, we can obtain the dual of the optimization problem (2.8)

$$\begin{aligned} \gamma^{*-2} \quad &= \quad \max_{\boldsymbol{\lambda}} \sum_{i=1}^{n} \lambda_i - \frac{1}{2\beta} \sum_{i=1}^{n} \sum_{j=1}^{n} \lambda_i \lambda_j y^i y^j (\mathbf{x}^i \cdot \mathbf{x}^j) \\ & \quad \quad \text{subject to} \quad 0 \leq \boldsymbol{\lambda} \leq 1, \ \boldsymbol{\lambda}^\top \mathbf{y} = 0 \end{aligned} \tag{2.13}$$

which is also a quadratic program.

After getting the dual solution, one can easily recover $\mathbf{w}$ by (2.10) although the primal solution $\mathbf{w}$ is not needed to define the resulting classifier. Instead, one can express the classifier strictly in terms of the dual solution $\boldsymbol{\lambda}$ and the offset b.

First, to compute b, notice that the following KKT (Karush-Kuhn-Tucker) condition is satisfied at the solution

$$\frac{\partial L}{\partial \xi_i} = 1 - \lambda_i - \mu_i = 0 \tag{2.14}$$

$$\mu_i \geq 0 \tag{2.15}$$

$$\lambda_i [y^i (\mathbf{x}^i \cdot \mathbf{w} + b) - 1 + \xi_i] = 0 \tag{2.16}$$

$$\mu_i \xi_i = 0 \tag{2.17}$$

Note that we can replace $\mathbf{w}$ in (2.16) by (2.10) and therefore express it strictly in terms of λ, without requiring $\mathbf{w}$. As a result, for the data points $\mathbf{x}^i$ such that $\lambda_i < 1$, we have $\mu_i > 0$ and $\xi_i = 0$ from (2.14) and (2.17). Therefore we can take the data points for which $0 < \lambda_i < 1$ (and thus $\xi_i = 0$) and use (2.16) to compute b.

After solving the optimization problem (2.13) and recovering b, a new data point $\mathbf{x}_{new}$ can then be classified according to the classification rule

$$\hat{y}_{new} = \operatorname{sgn}\left(\frac{1}{\beta} \sum_{i=1}^{n} \lambda_i y^i (\mathbf{x}^i \cdot \mathbf{x}_{new}) + b \right) \tag{2.18}$$

Nonlinear support vector machines

Furthermore, the nonlinear structure of data can be explored when linear discriminant does not perform well. Specifically, we can project a data point $\mathbf{x}$ in $\mathbb{R}^d$ to a high dimensional feature space $\mathbb{R}^F$, by a nonlinear mapping function $\boldsymbol{\phi} : \mathbb{R}^d \to \mathbb{R}^F, F > d$, and proceed to training and testing in the feature space.

Here we can take advantage of the fact that for the dual SVM, during the training process (2.13) or during classification (2.18), only the inner products between data points are required. This fact implies that we only need to choose a function $K : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$

$$K(\mathbf{x}, \mathbf{x}') = \boldsymbol{\phi}(\mathbf{x}) \cdot \boldsymbol{\phi}(\mathbf{x}') \tag{2.19}$$

instead of defining $\boldsymbol{\phi}(\mathbf{x})$ explicitly. The ability to use only inner products is an important property that makes SVM training and testing procedures possible, even in very high-dimensional feature spaces, and applicable to classifying objects other than vectors, such as graphs and strings (Shawe-Taylor and Cristianini, 2004; Schoelkopf and Smola, 2002).

The function K is called the kernel function. Given the training set $\mathbf{x}^1, ...\mathbf{x}^n$, the $n \times n$ matrix K formed by $K_{ij} = K(\mathbf{x}^i, \mathbf{x}^j)$ is called the kernel matrix. A natural question to ask is, given a function $K(\mathbf{x}, \mathbf{x}')$, how to decide whether it is a kernel function, or whether there exists a function $\boldsymbol{\phi}(\mathbf{x})$ such that $K(\mathbf{x}, \mathbf{x}') = \boldsymbol{\phi}(\mathbf{x}) \cdot \boldsymbol{\phi}(\mathbf{x}')$. The answer is provided by the following lemma.

Lemma 1 *(Shawe-Taylor and Cristianini, 2004; Schoelkopf and Smola, 2002) A function $K : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ can be decomposed as an inner product for some feature map $\boldsymbol{\phi}$*

$$K(\mathbf{x}, \mathbf{x}') = \boldsymbol{\phi}(\mathbf{x}) \cdot \boldsymbol{\phi}(\mathbf{x}')$$

if and only if the function is symmetric and the matrix formed by restriction to any subset of the space $\mathbb{R}^n$ is positive semi-definite.[3]

An example of kernel functions is the polynomial kernel $K(\mathbf{x}, \mathbf{x}') = (\mathbf{x} \cdot \mathbf{x}')^p$ where p is the polynomial degree. Another widely-used kernel that I will make use of below is the

[3]An $n \times n$ matrix X is positive semi-definite if X is symmetric and $\mathbf{z}^\top X \mathbf{z} \geq 0$ for all vectors $\mathbf{z}$ with length n.

radial basis function (RBF) kernel (Shawe-Taylor and Cristianini, 2004; Burges, 1998)

$$K(\mathbf{x}, \mathbf{x}') = \exp\left(-\frac{\|\mathbf{x} - \mathbf{x}'\|^2}{\sigma^2}\right) \tag{2.20}$$

where σ is a width controlling parameter. The RBF kernel implies a function $\boldsymbol{\phi}(\mathbf{x})$ that is infinite dimensional.

With the introduction of the kernel matrix K, the dual SVM (2.13) can be reformulated as

$$\begin{aligned} \gamma^{*-2} &= \max_{\boldsymbol{\lambda}} \sum_{i=1}^{n} \lambda_i - \frac{1}{2\beta} \sum_{i=1}^{n} \sum_{j=1}^{n} \lambda_i \lambda_j y^i y^j K_{ij} \\ &= \max_{\boldsymbol{\lambda}} \boldsymbol{\lambda}^\top \mathbf{e} - \frac{1}{2\beta} \langle K \circ \boldsymbol{\lambda}\boldsymbol{\lambda}^\top, \mathbf{y}\mathbf{y}^\top \rangle \\ &\text{subject to} \quad 0 \leq \boldsymbol{\lambda} \leq 1, \ \boldsymbol{\lambda}^\top \mathbf{y} = 0 \end{aligned} \tag{2.21}$$

where $A \circ B$ produces a matrix the same size as A and B such that $(A \circ B)_{ij} = A_{ij} B_{ij}$, and $\langle A, B \rangle$ is the inner product of the two matrices A and B given by $\langle A, B \rangle = trace(B^\top A) = \sum_{ij} A_{ij} B_{ij}$. Now a new data point $\mathbf{x}_{new}$ will be classified by

$$\hat{y}_{new} = \operatorname{sgn}\left(\frac{1}{\beta} \sum_{i=1}^{n} \lambda_i y_i K(\mathbf{x}^i, \mathbf{x}_{new}) + b\right). \tag{2.22}$$

Generalization error

Given a classification algorithm, one is typically interested in the statistical stability and robustness of the algorithm. In fact, for support vector machines there is a simple theoretical justification that depends only on the expected number of support patterns. If an SVM is trained on $n-1$ out of n examples, leaving one example out as testing data, a bound on the expected misclassification error of support vector machines is given by (Vapnik, 1995)

$$E[p(error)] \leq \frac{E[\text{number of support vectors}]}{\text{number of training samples}}$$

where $p(error)$ is the probability of error, or risk, of an SVM trained on $n-1$ examples, $E[p(error)]$ is the expectation of the risk over all possible choices of training set with size $n-1$. Here E[number of support vectors] is the expectation of the number of support vectors over all possible choices of training sets with size n. This bound suggests that when the number of support vectors is small, the risk of a support vector machine will be low.

Beyond this simple argument, there exists a large body of work analyzing the generalization performance of supervised SVMs (Cristianini and Shawe-Taylor, 2000; Shawe-Taylor et al., 1998; Vapnik, 1998; Shawe-Taylor et al., 1996; Vapnik, 1995, 1982). It has been shown that a large margin indicates good generalization performance. These theoretical analyses help justify the basic principle of support vector machines, and moreover can help one design SVM training and testing procedures, as well as select parameters. Later on, I will exploit this large margin principle for unsupervised training.

2.1.2 Multi-class Support Vector Machines

So far I have only been considering binary classification problems, where the class labels can only take values from $\{-1, +1\}$. However, in most practical situations, there are more than two classes. In this section, I review standard methods for multi-class support vector machines including the specific techniques I will use in Chapter 4 below.

One versus the rest classification

To perform κ-class classification, the idea behind "one versus the rest" classification is to construct κ support vector machines, where each is trained to separate one class from the others. For each classifier $f(\boldsymbol{\lambda}^j, b^j)$, take the output as $\frac{1}{\beta}\sum_{i=1}^{n} \lambda_i^j y^i K(\mathbf{x}^i, \mathbf{x}_{new}) + b^j$ where $j = 1, ..., \kappa$. Then the class label is determined by the maximum output among the κ classifiers.

The main problem with this method is that the κ classifiers are trained on κ different binary problems, and it is not clear that the real-valued outputs of different classifiers are comparable. The other problem is that during training, each classifier typically encounters many more negative than positive training examples, which makes the class distribution unbalanced.

All pairs classification

In the "all pairs" classification approach, a classifier is trained to distinguish each pair of classes. This results in $\kappa(\kappa-1)/2$ binary classifiers for κ classes, where each single classifier is trained on a smaller training set. The class label of a test example can then be determined by the class that gets the highest number of votes.

The problem with this method is that the number of classifiers to be trained is quite large. Although the training set for each classifier is smaller, when κ is large, training $\kappa(\kappa-1)/2$ classifiers can be more expensive than the one-versus-the-rest SVM.

Error-correcting output coding

Error-correcting output coding was first developed in (Dietterich and Bakiri, 1995). The main idea of this method is to generate a set of binary classifiers $f^1, ..., f^l$ such that the vector of their responses will completely determine the class label of an example, but do so in a way that is robust to misclassification errors of individual classifiers. In other words, each class is associated with a code vector in $\{-1, +1\}^l$. For κ classes, these vectors form a matrix $R \in \{-1, +1\}^{\kappa \times l}$, referred to as a decoding matrix. Given a new test example, one can first get the response vector produced by the set of binary classifiers, and then find the closest match between this vector and the rows of the decoding matrix. Here the "closeness" between two vectors is determined by the Hamming distance (the number of entries where the two vectors differ). This results in a multi-class classifier.

Error-correcting output coding has proved to be a successful method in many multi-class tasks. In (Allwein et al., 2000), a coding scheme tailored to large margin classifiers is developed, where Hamming-based decoding is replaced by a more sophisticated scheme taking the margin into account.

Multi-class objective and constraints

Unlike the methods above, recent multi-class SVM algorithms have been developed which have the goal of solving the problem directly (Crammer and Singer, 2001; Weston and

Watkins, 1999; Vapnik, 1998). This is achieved by modifying the SVM objective function as well as the constraints in a way to simultaneously allow the computation of multi-class classifier. In this section, we will focus our discussion on the approach proposed in (Crammer and Singer, 2001), which is the one I will adopt in my own work below.

Given training examples $(\mathbf{x}^1, y^1), ..., (\mathbf{x}^n, y^n)$, where each class label y^i is an integer from $\{1, ..., \kappa\}$, we need to extend the feature functions $\boldsymbol{\phi}(\mathbf{x}, y)$ to include the y-labels explicitly. The extended feature functions can be viewed as a concatenation of the underlying input features $\boldsymbol{\phi}(\mathbf{x})$, with one copy switched on at a time, according to the class label y

$$\boldsymbol{\phi}(\mathbf{x}, y) \quad = \quad \begin{bmatrix} \boldsymbol{\phi}(\mathbf{x}) \cdot \delta_{y,1} \\ \vdots \\ \boldsymbol{\phi}(\mathbf{x}) \cdot \delta_{y,\kappa} \end{bmatrix}$$

where $\delta_{a,b}$ is 1 if $a = b$ and 0 otherwise. Thus, each class r has its own separate weight vector $\mathbf{w}_r$, and these are concatenated into a single large weight vector $\mathbf{w} = (\mathbf{w}_1...\mathbf{w}_\kappa)$. Once a complete weight vector has been learned, subsequent test examples $\mathbf{x}$ are classified according to

$$y^* = \arg\max_y \ \mathbf{w} \cdot \boldsymbol{\phi}(\mathbf{x}, y)$$

The inner product of $\mathbf{w}$ and $\boldsymbol{\phi}(\mathbf{x}, r)$ is sometimes called the *similarity score* or *response* for class r.

The classification rule above will assign a data point to a class with the highest similarity score. The goal of the training algorithm is to find a vector $\mathbf{w}$ that results in small empirical error on the training set. To achieve this, similar to the binary case, given all training examples, we want to maximize a margin γ such that the similarity score of each feature vector to the true class is larger than those of the rest of the classes by at least γ:

$$\mathbf{w} \cdot \big(\boldsymbol{\phi}(\mathbf{x}^i, y^i) - \boldsymbol{\phi}(\mathbf{x}^i, r)\big) \geq \gamma(1 - \delta_{y^i,r}) \quad \forall i, r \tag{2.23}$$

If a training set is linearly separable, there always exists a vector $\mathbf{w}$ such that the above constraints are satisfied, since the differences between $\mathbf{w} \cdot \boldsymbol{\phi}(\mathbf{x}^i, y^i)$ and $\mathbf{w} \cdot \boldsymbol{\phi}(\mathbf{x}^i, r)$ can always be made arbitrarily large by increasing the norm of $\mathbf{w}$. Therefore, we need to control the values in $\mathbf{w}$ or else they could become unbounded. Given a training set that is linearly separable, we want to find a vector $\mathbf{w}$ of a small norm that satisfies (2.23), this results in an optimization problem

$$\max_{\mathbf{w},\gamma} \gamma \quad \text{subject to } \|\mathbf{w}\| \leq 1, \;\; \mathbf{w} \cdot \big(\boldsymbol{\phi}(\mathbf{x}^i, y^i) - \boldsymbol{\phi}(\mathbf{x}^i, r)\big) \geq \gamma(1 - \delta_{y^i,r}) \quad \forall i, r \tag{2.24}$$

which is equivalent to

$$\begin{aligned} & \max_{\mathbf{w},\gamma} \gamma \quad \text{subject to } \mathbf{w} \cdot \big(\boldsymbol{\phi}(\mathbf{x}^i, y^i) - \boldsymbol{\phi}(\mathbf{x}^i, r)\big) \geq \gamma\|\mathbf{w}\|(1 - \delta_{y^i,r}) \quad \forall i, r \\ = \; & \max_{\mathbf{w}} \frac{1}{\|\mathbf{w}\|} \quad \text{subject to } \mathbf{w} \cdot \big(\boldsymbol{\phi}(\mathbf{x}^i, y^i) - \boldsymbol{\phi}(\mathbf{x}^i, r)\big) \geq (1 - \delta_{y^i,r}) \quad \forall i, r \end{aligned} \tag{2.25}$$

Furthermore, (2.25) is equivalent to

$$\min_{\mathbf{w}} \frac{1}{2}\|\mathbf{w}\|^2 \quad \text{subject to } \mathbf{w} \cdot \big(\boldsymbol{\phi}(\mathbf{x}^i, y^i) - \boldsymbol{\phi}(\mathbf{x}^i, r)\big) + \delta_{y^i,r} \geq 1 \quad \forall i, r \tag{2.26}$$

In the more general case when the data is not separable, slack variables are introduced to cope with noisy examples, and the optimization problem (2.26) is modified as

$$\begin{aligned} \omega = \; & \min_{\mathbf{w},\boldsymbol{\xi}} \frac{1}{2}\beta\|\mathbf{w}\|^2 + \sum_{i=1}^{n} \xi_i \\ & \text{subject to} \quad \mathbf{w} \cdot (\boldsymbol{\phi}(\mathbf{x}^i, y^i) - \boldsymbol{\phi}(\mathbf{x}^i, r)) + \delta_{y^i,r} \geq 1 - \xi_i \quad \forall i, r \\ & \qquad\qquad\quad \xi_i \geq 0 \quad \forall i \end{aligned} \tag{2.27}$$

where β is the slack parameter, and ω is the margin loss, which is the multi-class analog of the inverse squared margin γ^{*-2} (2.8) in the two-class case.

The dual of (2.27) can be derived as:

$$\begin{aligned} & \max_{\boldsymbol{\Lambda}} \sum_{ir} \Lambda_{ir}(1 - \delta_{y^i,r}) - \frac{1}{2\beta}\left\|\sum_{ir} \Lambda_{ir}\left(\boldsymbol{\phi}(\mathbf{x}^i, y^i) - \boldsymbol{\phi}(\mathbf{x}^i, r)\right)\right\|^2 \\ & \qquad\qquad \text{subject to} \quad \sum_{r} \Lambda_{ir} = 1, \forall i \text{ and } \Lambda_{ir} \geq 0, \forall i, r \end{aligned} \tag{2.28}$$

where Λ is a $n \times \kappa$ matrix. (See Theorem 2 in Chapter 4 for more details.) This optimization problem is a quadratic program that can be solved in polynomial time.

This development yields an algorithm for training a multi-class SVM directly and efficiently. Compared to the multi-class training algorithms discussed previously, this method is elegant for its compactness. As a side-effect, it considers the entire problem (all data points and all classes) simultaneously. In (Crammer and Singer, 2001), some algorithmic improvements have been proposed to make the training technique practical for large-scale problems. My work in Chapter 4 extends this multi-class SVM formulation to the unsupervised and semi-supervised cases.

2.2 Unsupervised Classification and Clustering

One of the main contributions of this thesis is to develop *unsupervised* training algorithms for SVMs, that is, training algorithms for SVMs that do not require any training labels y. Unsupervised training of a classifier is equivalent to a strong form of *clustering*: the learned classifier proposes a label for each training example, and hence clusters the training data, but furthermore proposes a classification rule for the entire domain, including out-of-sample test points. Thus, in this way unsupervised classifier learning is more general than clustering. However, unsupervised classification learning techniques, like the unsupervised SVM algorithms I develop below, can still be compared to classical clustering algorithms if one restricts attention to the training data points only. Since generally I have to compare my results to clustering algorithms in chapters 3 and 4, I will briefly survey significant clustering algorithms here, concentrating primarily on algorithms that produce non-hierarchical clusterings (since hierarchical classification is not a topic covered in this thesis), and algorithms that can use different "similarity/dissimilarity" measures.

Clustering is one of the oldest forms of machine learning. It can be thought of as learning a classification labeling over a fixed set of data, except that no labels are available during training. Intuitively, the goal of clustering is to group the data objects into subsets or "clusters", so that the objects within each cluster are more closely related to each other than those in different clusters.

Once we describe the goal of clustering as organizing unlabeled data objects into groups whose members are similar in some way, the measure of "similarity/dissimilarity" used turns out to be fundamental in the whole process (Hastie et al., 2001; Duda and Hart, 1973). Using such a measure, one would expect that the distance between the objects in the same cluster is significantly smaller than the distance between the objects in the different clusters. (Or, analogously, that the similarities between objects in the same cluster are significantly higher than the similarities between objects in different clusters.) However, the specific results of clustering depends heavily on which similarity/dissimilarity measure is used.

2.2.1 Similarity/Dissimilarity Measures

The most common approach normally first considered in clustering is to simply use (squared) Euclidean distance $d(\mathbf{x}, \mathbf{x}') = \|\mathbf{x} - \mathbf{x}'\|^2$ to measure the dissimilarity between data objects. The choice of squared Euclidean distance implies that the resulting clusters will be invariant to translations or rotations. However, they will not be invariant to transformations that distort distance relationships. For example, different axis scalings can lead to different clusterings. One way to avoid this problem is to normalize the data before clustering, which means making each attribute of the data has zero mean and unit variance. In general, a Mahalanobis distance could be used, where $d(\mathbf{x}, \mathbf{x}') = (\mathbf{x} - \mathbf{x}')^\top \Sigma^{-1} (\mathbf{x} - \mathbf{x}')$ and Σ is an arbitrary positive definite covariance matrix that could be estimated from data (Duda and Hart, 1973). However, naive rescalings and estimates of Σ might be inappropriate if the large variance of an attribute is due to the presence of clusters. In this case, domain knowledge will be important to identify which attributes have higher influence in defining object similarity.

Instead of defining distance measures directly, another common approach is to define a *similarity* measure $s(\mathbf{x}, \mathbf{x}')$ from which one can derive a distance. For example, the normalized inner product

$$s(\mathbf{x}, \mathbf{x}') = \frac{\mathbf{x}^\top \mathbf{x}'}{\|\mathbf{x}\| \|\mathbf{x}'\|} \tag{2.29}$$

which gives the cosine of the angle between $\mathbf{x}$ and $\mathbf{x}'$, is a popular choice to define the similarity of meaning in natural language. A similarity measure like this is similar to a *kernel*, as defined in (2.19). In fact, the cosine similarity (2.29) corresponds to an inner product between feature vectors

$$s(\mathbf{x}, \mathbf{x}') = \frac{\mathbf{x}}{\|\mathbf{x}\|} \cdot \frac{\mathbf{x}'}{\|\mathbf{x}'\|}$$

and thus by Lemma 1 *is* a kernel. Standard kernels can therefore be used as similarity measures for clustering. For example, beyond the cosine function, the RBF kernel introduced previously (2.20) is also a popular choice for a similarity measure in clustering algorithms.

Similarity measures and distances are obviously closely related to each other. For example, there is a direct relationship between *positive definite* similarity measures (i.e. similarity measures that are kernels) and *squared Euclidean distances.* By Lemma 1, any positive semidefinite similarity function $K(\mathbf{x}, \mathbf{x}')$ is a kernel and can be decomposed as an inner product $K(\mathbf{x}, \mathbf{x}') = \boldsymbol{\phi}(\mathbf{x}) \cdot \boldsymbol{\phi}(\mathbf{x}')$ for some feature map $\boldsymbol{\phi}$. Therefore, a corresponding squared Euclidean distance can be defined by

$$\begin{aligned} d(\mathbf{x}, \mathbf{x}') &= K(\mathbf{x}, \mathbf{x}) - 2K(\mathbf{x}, \mathbf{x}') + K(\mathbf{x}', \mathbf{x}') \\ &= [\boldsymbol{\phi}(\mathbf{x}) - \boldsymbol{\phi}(\mathbf{x}')] \cdot [\boldsymbol{\phi}(\mathbf{x}) - \boldsymbol{\phi}(\mathbf{x}')] \\ &= \|\boldsymbol{\phi}(\mathbf{x}) - \boldsymbol{\phi}(\mathbf{x}')\|^2 \end{aligned}$$

Similarly, given a squared Euclidean distance $d(\mathbf{x}, \mathbf{x}') = \|\boldsymbol{\phi}(\mathbf{x}) - \boldsymbol{\phi}(\mathbf{x}')\|$, a corresponding kernel can be defined by $K(\mathbf{x}, \mathbf{x}') = \boldsymbol{\phi}(\mathbf{x}) \cdot \boldsymbol{\phi}(\mathbf{x}')$.

Not all similarity measures used by clustering algorithms are necessarily positive semidefinite, however, and correspondingly, not all dissimilarity measures are squared Euclidean distances. For example, another popular class of similarity measures used in spectral clustering algorithms (which I will introduce in Section 2.2.3 below) is any symmetric function $s(\mathbf{x}, \mathbf{x}') = s(\mathbf{x}', \mathbf{x})$ that is *nonnegative*, i.e. such that $s(\mathbf{x}, \mathbf{x}') \geq 0$. This in general is not sufficient to ensure s is positive definite. Nevertheless, positive semidefinite similarities still tend to be common in practice, even in the spectral clustering methods I will review in Section 2.2.3. For these algorithms, one need only ensure that the similarities also remain nonnegative, which means that the cosine similarity (2.29) cannot be directly used, but the RBF kernel similarity (2.20) can be and often is used. Therefore, to facilitate a fair comparison between the unsupervised SVM algorithms I will develop in Chapter 3 and 4 which use kernels (positive semidefinite similarities), and spectral clustering methods (Section 2.2.3 below) which use nonnegative similarities, I will focus on similarity measures like the RBF kernel (2.20) that are *both* positive semidefinite and nonnegative.

2.2.2 Number of Clusters

In addition to similarity/dissimilarity measures, another important issue is determining how many classes the data objects should be grouped into. As mentioned above, clustering is unsupervised. Hence, the number of clusters, κ, in addition to the class labels of the data objects, is unknown. A clustering algorithm can be used to infer the class labels. However, for most clustering algorithms, the number of clusters has to be given *a priori*. Unfortunately, there is no general agreed upon way to find the optimal number of clusters for any given data set. One possible approach is to apply cross-validation: run the clustering algorithm many times with different values for κ, compare the results, and finally choose a best value according to a given criterion. But one has to be aware that as κ increases, the clustering error generally decreases monotonically, which implies the risk of overfitting. More recently, many investigators have begun to use nonparametric Bayesian approaches with a prior on the possible number of clusters $\kappa = 1, 2, 3....$ For example, Dirichlet processes could be used to specify a prior on the number of clusters without requiring an upper bound on the number (Neal, 2003). However, I have not addressed the problem of automatically choosing the number of clusters in this thesis, and in the following, I will discuss several clustering algorithms that unless specified, assume that κ is given beforehand.

2.2.3 Clustering Algorithms

Different clustering algorithms can be designed based on different criteria and perspectives. In this thesis, I will mainly focus on the partition-based techniques which group the data directly into a flat clustering. This is usually achieved by optimizing a criterion function defined on the data set. Among partition-based clustering algorithms, there are many approaches, including k-means based clustering and graph cut based clustering. Below I will discuss two major classes of partition-based clustering algorithms—k-means based and graph cut based—respectively.

K-means clustering

K-means is still one of the most popular clustering techniques in use today (MacQueen, 1967). This method classifies a given data set into κ clusters by an iterative minimization of a within class dissimilarity criterion. For k-means clustering, squared Euclidean distance is normally chosen as the dissimilarity measure, although, in general, any distance measure $d(\mathbf{x}, \mathbf{x}')$ could be used.

Given a set of unlabeled data points $\mathbf{x}^1, ..., \mathbf{x}^n$, the main idea of k-means is to define κ *centers*, $\mathbf{m}_1, ..., \mathbf{m}_\kappa$, one for each cluster, with the goal of minimizing the sum of distances to the class centers to which data points are assigned. That is, one minimizes the objective

$$\min_{\mathbf{m}_1,...,\mathbf{m}_\kappa;\, C_1,...,C_n} \sum_{j=1}^{\kappa} \sum_{i=1, C_i=j}^{n} d(\mathbf{x}^i, \mathbf{m}_j) \tag{2.30}$$

where typically $d(\mathbf{x}^i, \mathbf{m}_j) = \|\mathbf{x}^i - \mathbf{m}_j\|^2$ and $C_i = j$ means the ith data point belongs to cluster j. In this optimization problem, both $C_1, ..., C_n$ and $\mathbf{m}_1, ..., \mathbf{m}_\kappa$ are unknown. To solve the problem, an iterative descent algorithm given below is used:

1. Randomly choose κ points as initial cluster centers.

2. Given the κ centers, minimize (2.30) by assigning each object to the closest cluster center; that is,

$$C_i = \arg \min_{j=1,...\kappa} d(\mathbf{x}^i, \mathbf{m}_j).$$

3. When all data points have been assigned, update the class centers $\mathbf{m}_1, ..., \mathbf{m}_\kappa$. If $d(\mathbf{x}^i, \mathbf{m}_j)$ equals squared Euclidean distance in particular, the cluster centers can be set to the *means* of the currently assigned clusters. In general, the centers can be assigned to the data points in each cluster that minimize some measure of the within-cluster distances (for example, either a sum or a maximum).

4. Iterate steps 2 and 3 until the assignments do not change.

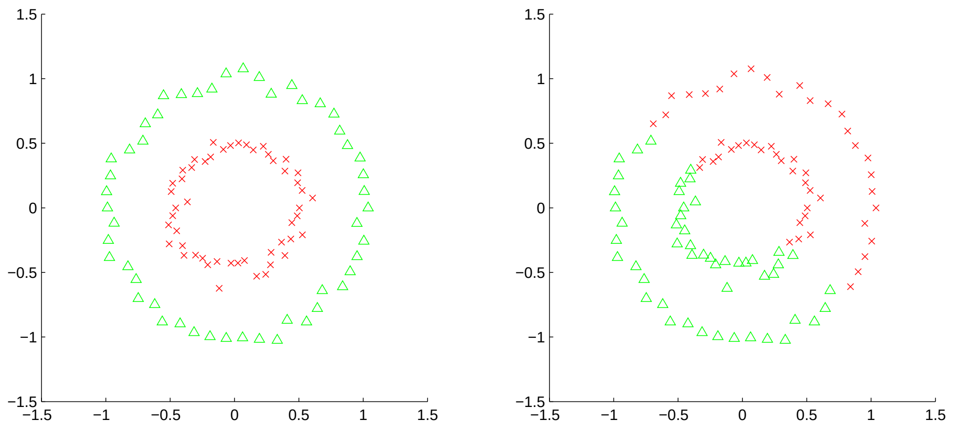


Figure 2.3: Simulated data: two circles. On the right are the clusters produced by k-means.

When using squared Euclidean distances and class means as centers, the convergence of this algorithm is guaranteed because each step of 2 and 3 reduces the objective value in (2.30). However, the result may be a local minimum, which implies that running the algorithm repeatedly can produce different results. To alleviate the effect of local minima, one typically runs the algorithm many times with different initial means, and choose the solution with the smallest objective value.

Although using squared Euclidean distance provides a nice (local) convergence guarantee, one problem this distance creates is that, in Step 3, “assigning each object to the closest cluster mean” implies that the data objects will be linearly clustered. This is a serious limitation when the natural clusters cannot be linearly separated; see for example Figure 2.3. For data with nonlinear structure such as this, an alternative, nonlinear distance measure must be used to capture the local structure in the data. The possibility of using alternative distance/similarity measures is best captured by spectral clustering techniques, which have become very popular in the past few years.

Spectral Clustering

A promising alternative to linear clustering methods is graph cut based (spectral) clustering, which clusters data by using the eigenvectors of a matrix generated in a graph theoretic formulation. Graph cut (or spectral) clustering methods have been empirically successful in many application areas and are now one of the dominant clustering algorithms used in machine learning (Shi and Malik, 2000, 1997; Hagen and Kahng, 1992). However, there are a variety of algorithms that use the the eigenvectors and generate the matrices in slightly different ways.

Before we can derive this class of algorithms, we need to define a weighted undirected graph $G = (V, E)$ on the data, where the nodes of the graph are the data objects to be clustered, and there is an edge between every pair of nodes. The weight on each edge $s(u, v)$, is a nonnegative similarity function between nodes u and v. If $\mathbf{x}^u$ and $\mathbf{x}^v$ are the corresponding data points of nodes u and v, a popular choice of the weight function is the RBF kernel we considered earlier (2.20)

$$s(u, v) = \begin{cases} \exp(\frac{-\|\mathbf{x}^u - \mathbf{x}^v\|^2}{\sigma^2}) & \text{if } u \neq v \\ 0 & \text{otherwise} \end{cases} .$$

Two-way spectral clustering

In two-way clustering, a graph $G = (V, E)$ is partitioned into two disjoint sets A and B, $A \cup B = V$, $A \cap B = \varnothing$, by removing edges connecting the two sets. This partitioning can be described by a labeling function y that indicates the two disjoint sets

$$y(u) = \begin{cases} 1 & \text{if } u \in A \\ -1 & \text{if } u \in B \end{cases} ,$$

and the degree of dissimilarity can be evaluated by the sum of weights (similarities) of the edges that have been removed by the labeling (i.e. those edges that connect nodes with different labels). In graph theory, this is called the *cut*:

$$cut(A, B) = \sum_{u \in A, v \in B} s(u, v). \tag{2.31}$$

Let $r(u) = \sum_v s(u, v)$ be the total connection weight from node u to the other nodes in the graph, and let R be a diagonal matrix with $\mathbf{r}$ on its diagonal. Then the cut cost can be rewritten as

$$\begin{aligned}
cut(A,B) &= \frac{1}{4}\sum_{u\in A, v\in B} s(u,v)(y(u)-y(v))^2 \\
&= \frac{1}{8}\sum_{(u,v)\in E} s(u,v)(y(u)-y(v))^2 \\
&= \frac{1}{4}\sum_{u,v} s(u,v)(1-y(u)y(v)) \\
&= \frac{1}{4}(\mathbf{y}^\top R\mathbf{y} - \mathbf{y}^\top S\mathbf{y}) \\
&= \frac{1}{4}\mathbf{y}^\top (R-S)\mathbf{y} \qquad (2.32)
\end{aligned}$$

where the matrix $R-S$ is usually known as the Laplacian matrix. Interestingly, the Laplacian matrix is guaranteed to be positive semidefinite and therefore defines a derived kernel similarity between data points (Fiedler, 1975, 1973).

Naturally, people consider partitioning the data by minimizing the cut value, this is the criterion called *minimum cut*. Although there are exponential number of possible partitions, finding the minimum cut of a graph is a well-studied problem and there exist efficient algorithms to solve it (Papadimitriou and Steiglitz, 1998). However, without introducing any other constraint, the minimum cut criterion favors cutting small pieces of isolated nodes in the graph. This will be undesirable in most cases. To overcome this problem, the sizes of A and B should be taken into account during the process of clustering.

Ratio cut The ratio cut algorithm (Hagen and Kahng, 1992) introduces the sizes of A and B with attempt to minimize the cut cost while keeping the partitions balanced simultaneously. The ratio cut objective is

$$Rcut(A,B) = \frac{cut(A,B)}{|A|} + \frac{cut(A,B)}{|B|} \qquad (2.33)$$

where $|\cdot|$ denotes the size of a set.

Let $\alpha = |A|/|G|$. Then the ratio cut objective can be reformulated as

$$Rcut(A,B) = \frac{1}{8|G|}\sum_{u,v} s(u,v)(y(u)-y(v))^2\left(\frac{1}{\alpha}+\frac{1}{1-\alpha}\right)$$

$$= \frac{\sum_{u,v} s(u,v)(y(u)-y(v))^2}{8\alpha(1-\alpha)\sum_u y^2(u)}.$$

If we define a new function f

$$f(u) = y(u) + (1-2\alpha) = \begin{cases} 2(1-\alpha) & \text{if } u \in A \\ -2\alpha & \text{if } u \in B \end{cases}, \tag{2.34}$$

it is easy to show that

$$f(u) - f(v) = y(u) - y(v)$$

and

$$\begin{aligned} \sum_u f^2(u) &= \sum_{u\in A} 4(1-\alpha)^2 y^2(u) + \sum_{u \in B} 4\alpha^2 y^2(u) \\ &= 4(1-\alpha)^2\alpha \sum_u y^2(u) + 4\alpha^2(1-\alpha)\sum_u y^2(u) \\ &= 4\alpha(1-\alpha)\sum_u y^2(u). \end{aligned}$$

Consequently, $y(u)$ can be replaced by $f(u)$:

$$\begin{aligned} Rcut(A,B) &= \frac{\sum_{u,v} s(u,v)(f(u)-f(v))^2}{2\sum_u f^2(u)} \\ &= \frac{\mathbf{f}^\top (R-S)\mathbf{f}}{\mathbf{f}^\top \mathbf{f}}. \end{aligned} \tag{2.35}$$

At this point, we can see that the objective of ratio cut takes the form of the Rayleigh quotient (Golub and Van Loan, 1996). However, since the discrete optimization of the objective (2.33) is NP-hard (Hagen and Kahng, 1992), we need to relax the discrete constraints (2.34) on f. One way to do that is to simply give up the discreteness while still keeping the property that $\mathbf{f}^\top \mathbf{e} = 0$ to enforce the partitions are balanced. In this case, the ratio cut problem turns into solving the following optimization problem

$$\min \mathbf{f}^\top (R-S)\mathbf{f} \text{ subject to } \|\mathbf{f}\| = 1, \mathbf{f}^\top \mathbf{e} = 0 \tag{2.36}$$

This problem can be readily solved by computing the second smallest eigenvector of the Laplacian matrix $R - S$ (Golub and Van Loan, 1996), and the cluster labels can be decided by the signs of vector $\mathbf{f}$.

The ratio cut is superior to the minimum cut because it takes the sizes of the partitions into consideration. However, from the ratio cut objective in (2.33), we can see that the similarity within each group is ignored. In other words, the ratio cut is able to find a partitioning where the cut cost is small and the two parts are relatively balanced, but it might create a partition where the members within each subset are only weakly related.

Normalized cut The normalized cut algorithm (Shi and Malik, 1997) addresses the shortcomings of the ratio cut by computing the cut objective as a fraction of the total edge connections to all the nodes in the graph, instead of the size of each part. That is, we adopt a normalized cut criterion given by

$$Ncut(A,B) = \frac{cut(A,B)}{assoc(A,V)} + \frac{cut(A,B)}{assoc(B,V)} \tag{2.37}$$

where $assoc(A,V) = \sum_{u\in A, t\in V} s(u,t) = \sum_{u\in A} r(u)$ is the total edge weight from A to all the nodes in the graph. With this normalized cut objective, the normalized cut algorithm is able to group the data into clusters that are widely separated from each other, while also being tight within themselves.

Similar to normalized cut, there is a measure called normalized association:

$$Nassoc(A,B) = \frac{assoc(A,A)}{assoc(A,V)} + \frac{assoc(B,B)}{assoc(B,V)}$$

which reflects how tightly the nodes are related to each other in each group. Interestingly, it is easy to see that $Ncut(A,B) = 2 - Nassoc(A,B)$, which means minimizing normalized cut is equivalent to maximizing normalized association. This also shows evidence that the disassociation between groups and the association within groups are both taken into account in normalized cut.

To develop an efficient computational procedure, we can similarly rewrite the normalized cut objective through the following steps. First let $\beta = \frac{\sum_{u\in A} r(u)}{\sum_u r(u)}$, the normalized cut objective can then be written as

$$\begin{aligned} Ncut(A,B) &= \frac{1}{8\sum_u d(u)} \sum_{u,v} s(u,v)(y(u)-y(v))^2 \left(\frac{1}{\beta} + \frac{1}{1-\beta}\right) \\ &= \frac{\sum_{u,v} s(u,v)(y(u)-y(v))^2}{8\beta(1-\beta)\sum_u y^2(u)r(u)} \end{aligned}$$

then we also define a new function g

$$g(u) = y(u) + (1-2\beta) \quad = \quad \begin{cases} 2(1-\beta) & \text{if } u \in A \\ -2\beta & \text{if } u \in B \end{cases} \tag{2.38}$$

which has similar properties that

$$g(u) - g(v) = y(u) - y(v)$$

$$\sum_u g^2(u)r(u) = 4\beta(1-\beta)\sum_u y^2(u)r(u).$$

Thus, the normalized cut objective can be reformulated as

$$\begin{aligned} Ncut(A,B) &= \frac{\sum_{u,v} s(u,v)(g(u)-g(v))^2}{2\sum_u g^2(u)r(u)} \\ &= \frac{\mathbf{g}^\top(R-S)\mathbf{g}}{\mathbf{g}^\top R\mathbf{g}}. \end{aligned} \tag{2.39}$$

While minimizing the normalized cut objective, we still have the discrete constraint (2.38) on g, which causes NP-hardness (Shi and Malik, 1997). Again, this is relaxed by only keeping the property that $\mathbf{g}^\top R\mathbf{e} = 0$.

Thus, the normalized cut problem turns into solving for the second smallest eigenvector of the generalized eigenvalue system

$$(R-S)\mathbf{g} = \lambda R\mathbf{g} \tag{2.40}$$

To achieve this, one can transform the eigensystem into a standard one

$$R^{-\frac{1}{2}}(R-S)R^{-\frac{1}{2}}\mathbf{g}' = \lambda\mathbf{g}' \tag{2.41}$$

where $\mathbf{g}' = R^{\frac{1}{2}}\mathbf{g}$ is a normalization of $\mathbf{g}$. It can be easily shown that the second smallest eigenvector of (2.40) corresponds to that of (2.41). Therefore, we can solve the normalized cut problem by calculating the second smallest eigenvector of the matrix $R^{-\frac{1}{2}}(R-S)R^{-\frac{1}{2}}$.

Normalized cut is one of the most popular clustering algorithms used in machine learning today. It can use any nonnegative similarity measure as discussed in section 2.2.1, which gives the technique a lot of flexibility that can be exploited in practice. For example, by using the RBF kernel as the similarity measure, the simulated two circles data as shown in Figure 2.30 (left) can be easily separated by the normalized cut algorithm. Furthermore, the normalized cut objective has a nice theoretical property that it converges to some limit clustering as the sample size increases, which means the resulting clustering becomes more stable as the amount of data increases (von Luxburg and Ben-David, 2005; von Luxburg et al., 2004).

κ-way spectral clustering

In practice, it is important to be able to cluster data into κ classes where in general $\kappa > 2$. There are several possible ways to perform κ-way spectral clustering, where κ is the number of clusters. One technique is recursive two-way clustering, which requires repetitions of the eigenvector computation. An alternative is to use multiple eigenvectors simultaneously, which has been observed to be better at computing a κ-way partitioning experimentally (Alpert et al., 1999).

The framework of the latter method is to first embed the data points in the space of the top κ eigenvectors, and then cluster the embedded points using another clustering algorithm, such as k-means (Ng et al., 2001). This type of methods is quite effective in finding multiple clusters of the data, and it is more efficient than the recursive algorithms.

Other clustering algorithms

Although I do not explicitly consider them in my thesis, there are a large number of other clustering algorithms that have been developed. Two prominent examples are hierarchical clustering and correlation clustering.

Hierarchical clustering Hierarchical clustering is among the best known clustering methods due to its conceptual simplicity. There are two different forms, known as agglomerative and divisive respectively. Agglomerative procedures start with n singleton clusters and proceed by merging existing clusters in each step until some termination criterion is reached. Divisive procedures, by contrast, start with all data objects in one cluster and proceed by splitting the clusters successively.

The primary criticism of hierarchical clustering is that it is not robust to outliers and noise. Once an object has been assigned to a cluster, the decision will not be reconsidered, even though it might continue to influence future partitions. Moreover, the computational complexity for hierarchical clustering is at least $O(n^2)$, which can limit its practicality for large-scale problems.

I do not consider hierarchical forms of unsupervised learning in my work below, and therefore do not consider comparison to hierarchical clustering in this thesis.[4]

Correlation clustering Correlation clustering is another graph theory based clustering problem (Bansal et al., 2002). Similar to spectral clustering, it takes a graph as input, but employs a simpler model where each edge is either labeled $+$ or $-$ to indicate whether a pair of nodes are similar or not. The goal is to partition the graph to maximize the *agreement*: the number of positive edges within clusters plus the number of negative edges between clusters. Interestingly, for correlation clustering, the number of clusters κ is not required a priori; it can be any number from 1 to n, whichever maximizes the agreement. Unfortunately, correlation clustering is an NP-hard problem (Bansal et al., 2002). In (Bansal et al., 2002), a PTAS (polynomial-time approximation scheme) is proposed for maximizing agreements. Later, a semidefinite relaxation for correlation clustering has been developed (Swamy, 2004), which is one of the recent applications of semidefinite programming techniques in machine learning.

Unfortunately this approach has not been applied in practice and requires a global tradeoff parameter λ to be set, which has a profound influence on the number of clusters chosen. I have therefore not compared to this approach in this thesis.

[4]However hierarchical unsupervised SVMs could be a topic for future work.

2.2.4 Evaluation of Clustering Performance

A related issue of designing a clustering algorithm is the evaluation of the resulting clusters. Unfortunately, there is no definitive way to measure the performance of a clustering algorithm. In my work, I have used an objective measure called mis-assignment rate. To evaluate a clustering algorithm, I take a set of *labeled* data, remove the labels, run the clustering algorithms, find the optimal correspondence between the true class labels and the resulting clusters, and then measure the number of misclassifications. This has the advantage of providing an objective assessment of clustering performance, but it relies on the assumption that the classes provided by the labeled data represent the true structure of the data that we are trying to recover. Below in chapters 3 and 4 I will use this technique to compare the quality of the unsupervised SVM algorithms I develop to standard k-means and spectral clustering algorithms.

2.3 Hidden Markov Models

My first set of thesis contributions (in chapters 3 and 4) concentrate on univariate classification, where decisions on a single class variable are made. Similarly, all of the learning techniques I have surveyed so far, whether supervised or unsupervised have also focused solely on making univariate class predictions.

However, many important problems require one to make sequential *multivariate* predictions, for example, as in speech recognition (Jelinek, 1998) or biological sequence analysis (Durbin et al., 1998). In Chapter 5 below I extend my unsupervised SVM training technique to the multivariate case, and consider unsupervised training of sequence labelers. Therefore I briefly survey existing approaches to learning structured predictors from data here. In particular, I have focused on learning sequence labeling models in my own work, and therefore focus attention on hidden Markov models (HMMs). Here I introduce the main representational ideas for HMMs and discuss both the standard supervised and unsupervised training algorithms that I will respectively extend and compare to in Chapter 5 below.

HMMs are by far the most important foundational approach taken to sequence labeling problems to date. HMMs are a special form of graphical model for sequence data of the

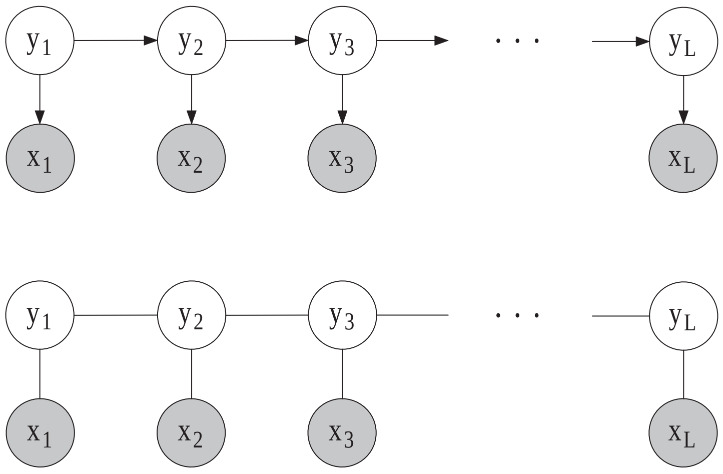


Figure 2.4: Equivalent directed and undirected representations of a hidden Markov model. Each vertical slice represents a time step. The top node in each slice is the state variable while the bottom node is the observation (or output) variable.

form $\mathbf{x} = (x_1, ..., x_L)$ and $\mathbf{y} = (y_1, ..., y_L)$, where $\mathbf{x}$ is a vector of observations and $\mathbf{y}$ is a corresponding sequence of states. The model assumes that the observation x_k at time k is conditionally independent of all other variables given the state y_k at the same time, and moreover y_k is conditionally independent of all other variables given y_{k-1}, x_k, y_{k+1}; see Figure 2.4. Higher order Markov models are also common where the probability at time k depends not only on the states at $k - 1$ but also on the ones before it. For ease of exposition, I restrict attention to the first order case.

2.3.1 Parametrization

To parameterize an HMM formally, first we assume a *stationary* model where the transition and observation probabilities do not change as a function of time. Although sophisticated Markov models can deal with continuous variables and continuous time, we will restrict ourselves to the discrete case here, and for simplicity we assume finite alphabets for the state and observation variables. So given a finite set of states $S = \{s_1, ..., s_\kappa\}$ and a set of possible observations $O = \{o_1, ..., o_v\}$, an HMM can be parameterized by the following:

1. An unconditional distribution on the initial state, $p(y_1)$, represented by a $\kappa \times 1$ vector $\boldsymbol{\pi}$ with $\pi_i = p(y_1 = s_i)$, where κ is the number of states;

2. The observation probability $p(x_k|y_k)$ for each time step k, also called emission probability. This is denoted by a $\kappa \times v$ matrix $B = \{B_i(o_\ell)\}$ where $B_i(o_\ell) = p(x_k = o_\ell|y_k = s_i)$;

3. The transition probability from one state to the next state, $p(y_{k+1}|y_k)$, specified by a $\kappa \times \kappa$ matrix A with $A_{ij} = p(y_{k+1} = s_j|y_k = s_i)$. For the rest of the discussion, we use $\theta = (\boldsymbol{\pi}, A, B)$ to denote all the parameters.

Thus, the joint probability can now be formulated as a product of the local probabilities due to the conditional independencies specified above. Given a configuration $(x_1, x_2, ..., x_L, y_1, y_2, ..., y_L)$, the joint probability is:

$$\begin{aligned} p(\mathbf{x}, \mathbf{y}) &= p(y_1) \prod_{k=2}^{L} p(y_k|y_{k-1}) \prod_{k=1}^{L} p(x_k|y_k) \\ &= \prod_{k=1}^{L} p(y_k|y_{k-1}) p(x_k|y_k) \end{aligned}$$

if we assume that $p(y_1|y_0) = p(y_1)$. It can be further written as a log-linear model through the following:

$$\begin{aligned} p(\mathbf{x}, \mathbf{y}) &= \prod_{k=1}^{L} p(y_k|y_{k-1}) p(x_k|y_k) \\ &= \prod_{k=1}^{L} \left[\prod_{a,b \in S} p(a|b)^{\delta_{y_k y_{k-1}, ab}} \right] \left[\prod_{a \in S, c \in O} p(c|a)^{\delta_{y_k x_k, ac}} \right] \\ &= \exp \left(\sum_{k=1}^{L} \left[\sum_{ab} w_{ab} \delta_{y_k y_{k-1}, ab} + \sum_{ac} w_{ac} \delta_{y_k x_k, ac} \right] \right) \\ &= \exp \left(\sum_{k=1}^{L} \left[\sum_{ab} w_{ab} \phi_{ab}(y_k y_{k-1}) + \sum_{ac} w_{ac} \phi_{ac}(y_k x_k) \right] \right) \\ &= \exp \left(\sum_{k} \sum_{j} w_j \phi_j(x_k, y_k, y_{k-1}) \right) / Z_1 \end{aligned} \tag{2.42}$$

Similarly we have

$$\begin{aligned} p(\mathbf{y}|\mathbf{x}) &= p(\mathbf{x},\mathbf{y})/p(\mathbf{x}) \\ &= \exp\left(\sum_k \sum_j w_j \phi_j(x_k, y_k, y_{k-1})\right)/Z_2(\mathbf{x}) \end{aligned} \tag{2.43}$$

where Z_1 and Z_2 are the normalization factors.

Given the parametrization, we can now focus on the operations that one can perform with the models. Generally speaking, there are three fundamental problem for HMMs.

1. ***Evaluation***. Given an HMM and the *local* probabilities, determine the probability $p(\mathbf{x})$ of an observation sequence $\mathbf{x}$. In other words, compute the probability that the observed sequence was produced by the model. This can be done efficiently by a recursive algorithm that computes the probabilities time step by time step, with a time complexity linear in the sequence length.

2. ***Decoding***. Given an observation sequence $\mathbf{x}$ and an HMM with its local probabilities, determine the optimal hidden state sequence, $\mathbf{y}^* = \arg\max_{\mathbf{y}} p(\mathbf{y}|\mathbf{x})$. This is also called the inference problem, and it can be solved in linear time by using the Viterbi algorithm (Rabiner, 1989).

3. ***Learning/training***. Given a set of training data that is produced by an HMM, determine the parameters of the model. The problem can be either *supervised* where in the training data, not only the observation sequences but also the state sequences are given, or *unsupervised* where only observation sequences are provided. Below I will discuss these two types of learning problems in more detail.

2.3.2 Supervised HMM Training

There are several different criteria that one could use for supervised learning—the choice usually depends on the intended application. For example, one can set the parameters with the values that maximize the joint likelihood $\prod_{i=1}^{n} p_\theta(\mathbf{x}^i, \mathbf{y}^i)$, which can be easily done by setting the observation and transition probabilities to the observed frequency counts of each state→state, state→observation, and initial state patterns.

More often, however, one is more interested in learning a *conditional model* $p(\mathbf{y}|\mathbf{x})$ rather than a joint model $p(\mathbf{x}, \mathbf{y})$, because the conditional model is needed as a decoder in a subsequent application. That is, the conditional model serves as the structured predictor of a joint labeling $\mathbf{y}$ for an input sequence $\mathbf{x}$. For conditional models it has long been observed that discriminative (conditional likelihood) training is advantageous compared to joint (maximum likelihood) training. In fact, the significant recent progress on learning structured predictors has been based on developing training procedures that exploit discriminative criteria, such as conditional likelihood or margin loss; for example, as in conditional random fields (Lafferty et al., 2001), discriminative sequence training (Altun et al., 2003; Tsochantaridis et al., 2004), and maximum margin Markov networks (Taskar et al., 2003). The decoding accuracy achieved by these techniques generally exceeds that of the more straightforward maximum likelihood approaches. Below I will primarily focus on maximum margin Markov networks (M^{3}N) because this is the approach I extend to the unsupervised case in Chapter 5 below. The goal of M^{3}N training is to learn the HMM predictors under the multivariate margin loss formulation.

Maximum margin Markov networks

Large margin classification (SVMs) and graphical probability models have often been regarded as independent of each other, however, they can actually work nicely together (Altun et al., 2003; Tsochantaridis et al., 2004; Taskar et al., 2003). The key observation is that combining large margin training principles with graphical probability models retains the useful properties of both methods, such as the flexibility of SVMs with kernels, and the ability of probabilistic models to deal with structured data.

To establish the connection, consider the alternative representation of a graphical model (HMM) as a linear model expressed by features, as shown in (2.43). That is, as we have seen in (2.43), the conditional model is in the log-linear form such that

$$\log p(\mathbf{y}|\mathbf{x}) \;=\; \sum_k \sum_j w_j \phi_j(x_k, y_k, y_{k-1}) - A(\mathbf{w}, \mathbf{x})$$

$$\begin{aligned} &= \sum_k \mathbf{w} \cdot \boldsymbol{\phi}(x_k, y_k, y_{k-1}) - A(\mathbf{w}, \mathbf{x}) \\ &= \mathbf{w} \cdot \boldsymbol{\phi}(\mathbf{x}, \mathbf{y}) - A(\mathbf{w}, \mathbf{x}) \end{aligned} \tag{2.44}$$

where we define

$$\boldsymbol{\phi}(\mathbf{x}, \mathbf{y}) = \sum_k \boldsymbol{\phi}(x_k, y_k, y_{k-1}). \tag{2.45}$$

and

$$A(\mathbf{w}, \mathbf{x}) = \log \sum_{\mathbf{y}} \exp(\mathbf{w} \cdot \boldsymbol{\phi}(\mathbf{x}, \mathbf{y})) = \log\ Z_2(\mathbf{x})$$

Note that for *classification*, that is determining the optimal label sequence $\mathbf{y}^*$ for a *given* input sequence $\mathbf{x}$, the normalization term $A(\mathbf{w}, \mathbf{x})$ is constant and does not matter.

If we drop the normalization, and concentrate instead on the classification boundaries directly, (Altun et al., 2003; Tsochantaridis et al., 2004; Taskar et al., 2003) show how a large margin criterion can be used to obtain a supervised, discriminative training procedure for HMMs. Recall that the multi-class SVM algorithm (2.24) introduced in Section 2.1.2 strives to maximize the margin γ subject to the constraints that the similarity of the feature vector to the correct class is larger than those of the rest of the classes by at least γ. For convenience I repeat the training problem here

$$\max_{\mathbf{w},\gamma} \gamma \quad \text{subject to } \|\mathbf{w}\| \leq 1,\ \mathbf{w} \cdot \big(\boldsymbol{\phi}(\mathbf{x}^i, y^i) - \boldsymbol{\phi}(\mathbf{x}^i, r)\big) \geq \gamma(1 - \delta_{y^i,r}) \quad \forall i, r \tag{2.46}$$

This can be easily extended to the multivariate case as follows. Given labeled training sequences $(\mathbf{x}^1, \mathbf{y}^1), ..., (\mathbf{x}^n, \mathbf{y}^n)$, where each individual training sequence is of the form $(\mathbf{x}^i = (x^i_1, ..., x^i_L),\ \mathbf{y}^i = (y^i_1, ..., y^i_L))$, one can optimize the global prediction margin:

$$\max_{\mathbf{w},\gamma} \gamma \quad \text{subject to } \|\mathbf{w}\| \leq 1,\ \mathbf{w} \cdot \big(\boldsymbol{\phi}(\mathbf{x}^i, \mathbf{y}^i) - \boldsymbol{\phi}(\mathbf{x}^i, \mathbf{u})\big) \geq \gamma\Delta(\mathbf{u}, \mathbf{y}^i) \quad \forall i, \mathbf{u}$$

where $\Delta(\mathbf{u}, \mathbf{y}^i) = \sum_{k=1}^{L}(1 - \delta_{u_k, y^i_k})$ counts the disagreements between $\mathbf{u}$ and $\mathbf{y}^i$. Note that when generalizing the large margin approach to the multivariate case, the loss is usually augmented to the per-label loss $\Delta(\mathbf{u}, \mathbf{y}^i)$ instead of the 0-1 loss. The effect of this extension is to favor solutions that produce fewer misclassifications in the predicted state sequence.

In practice, the data is usually non-separable, therefore we introduce slack parameters and use the standard transformation to eliminate γ:

$$\min \frac{\beta}{2}\|\mathbf{w}\|^2+\sum_i \xi_i \quad \text{subject to } \mathbf{w}^\top\big(\boldsymbol{\phi}(\mathbf{x}^i,\mathbf{y}^i)-\boldsymbol{\phi}(\mathbf{x}^i,\mathbf{u})\big)\geq\Delta(\mathbf{u},\mathbf{y}^i)-\xi_i \quad \forall i,\mathbf{u} \tag{2.47}$$

This problem can be solved using quadratic programming. Its corresponding dual is

$$\max \sum_{i,\mathbf{u}} \lambda_i(\mathbf{u})\Delta(\mathbf{u},\mathbf{y}^i) - \frac{1}{2\beta}\left\|\sum_{i,\mathbf{u}}\lambda_i(\mathbf{u})\left(\boldsymbol{\phi}(\mathbf{x}^i,\mathbf{y}^i)-\boldsymbol{\phi}(\mathbf{x}^i,\mathbf{u})\right)\right\|^2$$
$$\text{subject to } \sum_{\mathbf{u}}\lambda_i(\mathbf{u})=1;\ \lambda_i(\mathbf{u})\geq 0 \quad \forall i,\mathbf{u} \tag{2.48}$$

At this point, the number of constraints in the primal QP (2.47), or correspondingly the number of variables in the dual (2.48) is exponential in the length of the label sequence L, which is intractable in practice. To reduce the computational expense of solving this problem, marginal dual variables are defined as follows (Taskar et al., 2003):

$$\begin{aligned}\nu_{ik}(uu') &= \sum_{\mathbf{u}\sim(uu',k)}\lambda_i(\mathbf{u}) \quad \forall k;\ \forall u,u';\ \forall i\\ \mu_{ik}(u) &= \sum_{\mathbf{u}\sim(u,k)}\lambda_i(\mathbf{u}) \quad \forall k;\ \forall u;\ \forall i\end{aligned}$$

where $\mathbf{u}\sim(uu',k)$ denotes a full assignment $\mathbf{u}$ which is partially consistent with assignments uu' on the pair $(k,k-1)$, similarly for $\mathbf{u}\sim(u,k)$. Here $\nu_{ik}(uu')$ and $\mu_{ik}(u)$ are related in the way that

$$\sum_{u'}\nu_{ik}(uu')=\mu_{ik}(u)$$

Moreover, since $\boldsymbol{\phi}(\mathbf{x}^i,\mathbf{y}^i)=\sum_k\boldsymbol{\phi}(x_k^i y_k^i y_{k-1}^i)$ and $\Delta(\mathbf{u},\mathbf{y}^i)=\sum_k\Delta(\mathbf{u}_k,y_k^i)$, the original dual QP can be factorized into

$$\begin{aligned}\omega(\mathbf{y}^1,\ldots,\mathbf{y}^n) &= \max_{\boldsymbol{\mu},\boldsymbol{\nu}}\sum_{i,k,u}\mu_{ik}(u)1_{(u\neq y_k^i)}\\ &\quad -\frac{1}{2\beta}\sum_{ij,k\ell,uu',vv'}\nu_{ik}(uu')\nu_{j\ell}(vv')\Delta\boldsymbol{\phi}_{ik}(uu')^\top\Delta\boldsymbol{\phi}_{j\ell}(vv')\end{aligned} \tag{2.49}$$

$$\begin{aligned}\text{subject to}\quad & \mu_{ik}(u) \geq 0, \forall iku \\ & \nu_{ik}(uu') \geq 0, \forall ikuu' \\ & \sum_{u'} \nu_{ik}(uu') = \mu_{ik}(u), \quad \sum_{u} \mu_{ik}(u) = 1\end{aligned}$$

where $\Delta\boldsymbol{\phi}_{ik}(uu') = \boldsymbol{\phi}(x^i_k y^i_k y^i_{k-1}) - \boldsymbol{\phi}(x^i_k uu')$. Here i, j index training cases, k, ℓ index locations in each training sequence, and u, u' index possible relabelings at these locations. There is a dual variable $\mu_{ik}(u)$ corresponding to each singleton relabeling, and a dual variable $\nu_{ik}(uu')$ corresponding to each adjacent pair relabeling.

This formulation provides a reduced, polynomial size QP with polynomial complexity. Note however that this is a *supervised* training algorithm, as it requires the sequence labels $\mathbf{y}^i$ to be provided for each input sequence $\mathbf{x}^i$. One of my main research contributions in this thesis is to extend this supervised HMM training algorithm to the unsupervised case (in Chapter 5 below).

2.3.3 Unsupervised HMM Training

Unsupervised training of HMMs, where one is given the observation sequences $\mathbf{x}^1, \mathbf{x}^2, ..., \mathbf{x}^n$ but *no* accompanying label sequences is a classical problem that has normally been tackled from a probabilistic perspective. (I will offer an alternative margin based perspective in Chapter 5 below.) In fact, historically for the unsupervised training of HMMs, most researchers back off to a joint model view, and try to maximize the marginal likelihood $\max_\theta \prod_i p_\theta(\mathbf{x}_i)$, and thus recover a conditional model as a side-effect of acquiring $p(\mathbf{x}, \mathbf{y})$. However, with no hidden state sequences given, it is generally not possible to solve the problem analytically. Instead, an EM algorithm for HMMs is used, which is sometimes known as the Baum-Welch algorithm. As a particular case of EM, the Baum-Welch algorithm starts with some initial values of the parameters. At each iteration, the Expectation-step computes the expected probability of all possible hidden state transitions, then the Maximization-step re-calculates the parameters to maximize the objective based on the expected values. This procedure is repeated until convergence is reached.

Below I will introduce a sequence of algorithms that constitute the main part of EM training for HMMs, which I will compare to my own work in Chapter 5 below.

The forward algorithm

Define $\alpha_k(i) = p(x_1, ..., x_k, y_k = s_i|\theta)$ as the probability of witnessing all the observations generated up to time step k and staying at state s_i at time k. These α's can be computed using the forward algorithm:

1. The probability of initially being in state s_i and generating x_1 is given by
   $\alpha_1(i) = \pi_i B_i(x_1)$.

2. We can then arrive at state s_i from any state s_j with probability A_{ji} and generate observation x_{k+1}, thus yielding the α probabilities:
   $\alpha_{k+1}(i) = B_i(x_{k+1}) \sum_j \alpha_k(j) A_{ji}$ for $1 \leq k \leq L$.

It is easy to see that given the model, $p(\mathbf{x}) = \sum_i \alpha_L(i)$. In other words, we can use the forward algorithm to solve the evaluation problem for HMMs.

The backward algorithm

Define $\beta_k(i) = p(x_{k+1}, ..., x_L|y_k = s_i, \theta)$ as the probability of generating all the observations after time k, given that the system stays at state s_i at time k. These β probabilities can be computed via the following process:

1. Each state is allowed to be a possible end state, thus
   $\beta_L(i) = 1$

2. Then x_{k+1} can be generated from any state s_j, thus yielding a recursive computation of the β probabilities:
   $\beta_k(i) = \sum_j \beta_{k+1}(j) A_{ij} B_j(x_{k+1})$ for $1 \leq k \leq L$.

Given these computations, it is also easy to see that $p(\mathbf{x}|\theta) = \sum_i \alpha_k(i)\beta_k(i)$ for any k.

The Baum-Welch algorithm

Finally, we can use these forward and backward probabilities to provide the Expectation step for the Baum-Welch algorithm used to learn a good set of model parameters θ from unlabeled data. To find $\theta^* = \arg\max_\theta p(\mathbf{x}|\theta)$, first define $\gamma_k(i) = p(y_k = s_i|\mathbf{x}, \theta)$ as the probability of the system being at state s_i at time k given the observations, and $\xi_k(i,j) = p(y_k = s_i, y_{k+1} = s_j|\mathbf{x}, \theta)$ as the transition probability from state s_i to s_j at time k given the observations. It is easy to see that $\gamma_k(i) = \sum_j \xi_k(i,j)$, and, for $k = 1, ...L$,

$$\gamma_k(i) = \frac{\alpha_k(i)\beta_k(i)}{\sum_j \alpha_k(j)\beta_k(j)} \tag{2.50}$$

For $k = 1, ...L-1$, $\xi_k(i,j)$ can be computed by

$$\xi_k(i,j) = \frac{\alpha_k(i)A_{ij}B_j(x_{k+1})\beta_{k+1}(j)}{\sum_j \alpha_k(j)\beta_k(j)} \tag{2.51}$$

$$= \frac{\gamma_k(i)A_{ij}B_j(x_{k+1})\beta_{k+1}(j)}{\beta_k(i)} \tag{2.52}$$

Formulas (2.50)-(2.52) provide the means to compute the expected probability of the hidden state transitions. Based on these expected values, we can re-estimate the parameters (the Maximization step) with a sequence of updates specified below until convergence:

$$\pi_i' = \gamma_1(i) \tag{2.53}$$

$$A_{ij}' = \frac{\sum_{k=1}^{L-1} \xi_k(i,j)}{\sum_j \sum_{k=1}^{L-1} \xi_k(i,j)} \tag{2.54}$$

$$B_i(o_\ell)' = \frac{\sum_{k=1, x_k = o_\ell}^{L} \gamma_k(i)}{\sum_{k=1}^{L} \gamma_k(i)} \tag{2.55}$$

Although EM has been widely used in many applications for unsupervised HMM training, there are several problems with it. First, EM fails to guarantee a global solution to the problem, therefore initialization is an important issue in practice. Second, as we have observed, EM takes a joint model view, therefore if we are interested in learning a discriminative model $p(\mathbf{y}|\mathbf{x})$, there is little reason to expect that the training criterion used by EM will recover a good decoder. One of the goals of my research in Chapter 5 is to mitigate both these problems by formulating a *convex* training objective that is also *discriminative* (i.e. it is based on maximizing a margin criterion).

2.4 Semidefinite Programming

Finally, one of the main algorithmic tools I exploit in the training algorithms I develop below is semidefinite programming (SDP). In particular, I will exploit semidefinite programming as a means for solving a relaxation of the combinatorial problem that arises from unsupervised SVM training (i.e. SVMs applied to clustering), as well as robust supervised SVM training.

Semidefinite programming is a recent extension of linear and quadratic programming that was significantly advanced by (Nesterov and Nemirovskii, 1994), which provided effective algorithmic foundations as well as polynomial run-time guarantees for solution techniques based on interior point methods. Semidefinite programming has since had a significant impact on the field of combinatorial optimization, and has opened up exciting new avenues of research in machine learning (Lanckriet et al., 2004; Swamy, 2004; Weinberger et al., 2004; Goemans and Williamson, 1995).

A semidefinite programming problem is a convex optimization problem where one optimizes a symmetric $n \times n$ matrix of variables X

$$\begin{aligned} \min_X \langle C, X \rangle \quad \text{subject to} \quad & \langle A_i, X \rangle = b_i, \;\; i = 1...m \\ & X \succeq 0 \end{aligned} \tag{2.56}$$

where $X \succeq 0$ denotes the constraint that X must remain positive semidefinite (Helmberg, 2000; Boyd and Vandenberghe, 2004). Here, the objective is linear in X, and the semidefinite constraint simply means that X must satisfy $\mathbf{z}^\top X \mathbf{z} \geq 0$ for any vector $\mathbf{z} \neq \mathbf{0}$, which implies infinitely many liner constraints. However, it is still a convex constraint on X. That is, for any two positive semidefinite matrices X and Y, any convex combination will be positive semidefinite, since $\mathbf{z}^\top(\rho X + (1-\rho)Y)\mathbf{z} = \rho \mathbf{z}^\top X \mathbf{z} + (1-\rho)\mathbf{z}^\top Y \mathbf{z} \geq 0$ for $0 \leq \rho \leq 1$.

The dual form of (2.56) is (Helmberg, 2000; Boyd and Vandenberghe, 2004):

$$\max_{\mathbf{y}, Z} \mathbf{b}^\top \mathbf{y} \quad \text{subject to} \quad Z + \textstyle\sum_i y_i A_i = C, \;\; Z \succeq 0$$

Semidefinite programming can be solved by algorithmic approaches in polynomial time. Among them, interior point methods (or "barrier") methods are currently one of the most effective approaches (Nesterov and Nemirovskii, 1994). The worst-case complexity of these methods is $O(m^2n^{2.5})$ in general, and one can exploit the structure of the problem (e.g. sparsity) for more computational efficiency. In fact, there exist many software packages for solving semidefinite programming problems using interior point methods, including SeDuMi (Sturm, 1999), SDPT3 (Toh et al., 1999) and CSDP. Although these techniques are not as well developed as methods for solving quadratic programs, progress is continuing and the current tools are starting to become adequate for solving practical problems.

I will generally exploit these existing tools in my own research below. That is, my thesis research focuses on applying semidefinite programming as a flexible new tool for deriving useful new machine learning algorithms. I do not, however, contribute any significant new ideas to the practice of solving general SDPs.

Chapter 3

Unsupervised Two-class Support Vector Machines

Clustering is one of the oldest form of machine learning. Nevertheless, it has received renewed attention recently with the advent of nonlinear clustering methods based on kernels. Kernel based clustering methods continue to have a significant impact on recent work in machine learning (Ng et al., 2001; Kandola et al., 2001), computer vision (Shi and Malik, 2000), and bioinformatics (Kluger et al., 2003). Although many variations of kernel based clustering has been proposed in the literature, most of these techniques share a common "spectral clustering" framework (outlined in Section 2.2.3) that follows a generic recipe: one first builds the kernel ("affinity") matrix, normalizes the kernel, performs dimensionality reduction, and finally clusters (partitions) the data based on the resulting representation (Weiss, 1999).

In this chapter, I propose a new approach to clustering, based on the large margin classification ideas presented in Section 2.1. The primary focus will be on the final partitioning step where the actual clustering occurs. Once the data has been preprocessed and a kernel matrix has been constructed (and its rank possibly reduced), many variants have been suggested in the literature for determining the final partitioning of the data. The predominant strategies include using k-means clustering (Ng et al., 2001), minimizing various forms of graph cut cost (Kandola et al., 2001) (relaxations of which amount to clustering based on eigenvectors (Weiss, 1999)), and finding strongly connected components in a Markov chain defined by the normalized kernel (Chennubhotla and Jepson, 2002). Some other recent alternatives are correlation clustering (Bansal et al., 2002) and support vector clustering (Ben-Hur et al., 2001).

Here I propose instead to use a maximum margin separting hyperplane to separate the data into clusters. One nice aspect of the approach that I propose is that it provides a simple unified view of unsupervised, supervised, and semi-supervised learning—a common unifying approach to these problems is lacking in most of the current literature on these topics. In particular, I start with a completely supervised learning technique—support vector machines (SVMs)—and modify it for unsupervised learning, with the goal of achieving a simple, unified way of solving a variety of problems, including unsupervised and semi-supervised learning.

3.1 Unsupervised Two-class SVMs

Recall that for *supervised* two class SVM training, we assume we are given labeled training examples $(\mathbf{x}^1, y^1), ..., (\mathbf{x}^n, y^n)$, where each example is assigned to one of the two classes $y^i \in \{-1, +1\}$. The goal of an SVM in this case is to find the linear discriminant that maximizes the minimum mislcassification margin, as shown in Figure 2.1. Furthermore, as shown in Section 2.1, this reduces to solving the optimization problem

$$\begin{aligned} \gamma^{*-2} \quad = \quad & \max_{\boldsymbol{\lambda}} \ \boldsymbol{\lambda}^\top \mathbf{e} - \frac{1}{2\beta}\langle K \circ \boldsymbol{\lambda}\boldsymbol{\lambda}^\top, \mathbf{y}\mathbf{y}^\top \rangle \\ & \text{subject to} \quad 0 \leq \boldsymbol{\lambda} \leq 1, \ \boldsymbol{\lambda}^\top \mathbf{y} = 0 \end{aligned} \tag{3.1}$$

However, in the *unsupervised* case we are not given the labels $\mathbf{y}$, and therefore cannot directly solve the supervised problem (3.1). Here I propose a "principle of (guarded) optimism" to convert a supervised machine learning algorithm to the unsupervised case: to optimize the parameters of the supervised model without training labels, first optimize the *labeling* (subject to constraints) so that the resulting supervised problem admits a good training outcome. This is a very general approach that, in principle, could be applied to any supervised learning algorithm to make it unsupervised. In this thesis, I will specifically consider applying this principle to support vector machines.

For unsupervised SVMs, instead of finding a large margin classifier given labels on the data, the idea is to find a *labeling* so that if one were to subsequently run an SVM training algorithm, the margin obtained would be maximal over all possible labelings. That is, given data $\mathbf{x}^1, .., \mathbf{x}^n$, we wish to assign the data points to two classes $y^i \in \{-1, +1\}$ so that the separation between the two classes is as wide as possible.

This appears to be a hard computational problem. Although I am not aware of any NP-hardness proof, previous investigators have suggested this as a training principle for semi-supervised SVMs, but have resorted to heuristic algorithms (Joachims, 1999) and branch and bound search (Vapnik, 1998). In my work, I have assumed the exact discrete problem is hard, and worked at deriving tractable approximations. Proving hardness remains an issue for future research.

Toward developing a tractable approximation, I show below that, with some derivation, one can re-express the problem so that it is convex (except for the integrality constraints), which suggests that there might be some hope of obtaining practical solutions. Subsequently, I relax the integer constraint to obtain a semidefinite program that yields soft cluster assignments which approximately maximize the margin between the classes. Therefore, one can obtain soft clusterings efficiently using widely available software.

Before proceeding with the main development, there are some preliminary issues I need to address. First, we clearly need to impose some sort of constraint on the class balance, since otherwise one could simply assign all the data points to the same class and obtain an unbounded margin. A related issue is that we would also like to avoid the problem of separating a single outlier (or very small group of outliers) from the rest of the data. Thus, to mitigate these effects I will impose a constraint that the difference in class sizes be bounded. This will turn out to be a natural constraint for semisupervised learning and is very easy to enforce. Second, one would like the clustering to behave gracefully on noisy data where the classes may in fact overlap, so I adopt the soft margin formulation of the maximum margin criterion. Finally, there is a small technical complication that arises with one of the SVM parameters: It turns out that an unfortunate nonconvexity problem arises when I include the use of the offset b in the underlying large margin classifier. I currently do not have a way to avoid this nonconvexity, and therefore I currently set $b = 0$ and therefore only consider homogeneous linear classifiers. The consequence of this restriction is that the constraint $\boldsymbol{\lambda}^\top \mathbf{y} = 0$ is removed from the dual SVM quadratic program (3.1). Although it would seem like this is a harsh restriction, the negative effects are mitigated by centering the data at the origin, which can always be imposed. Nevertheless, dropping this restriction remains an important issue. With these caveats in mind, I proceed to the main development.

3.2 Derivation of the Main Result

We face the following computational problem: we would like to solve for a labeling $\mathbf{y} \in \{-1,+1\}^n$ that leads to a maximum (soft) margin. Straightforwardly, one could attempt to tackle this optimization problem by directly formulating

$$\min_{\mathbf{y}\in\{-1,+1\}^n} \gamma^{*-2}(\mathbf{y}) \quad \text{subject to} \quad -\epsilon \le \mathbf{e}^\top \mathbf{y} \le \epsilon$$

where

$$\gamma^{*-2}(\mathbf{y}) = \max_{\boldsymbol{\lambda}} \ \boldsymbol{\lambda}^\top \mathbf{e} - \frac{1}{2\beta}\langle K \circ \boldsymbol{\lambda}\boldsymbol{\lambda}^\top, \mathbf{y}\mathbf{y}^\top \rangle \quad \text{subject to } 0 \le \boldsymbol{\lambda} \le 1$$

Unfortunately, $\gamma^{*-2}(\mathbf{y})$ is not a convex function of $\mathbf{y}$, and this formulation does not lead to an effective algorithmic approach. In fact, to obtain an efficient technique for solving this problem we need two key insights.

The first key step is to re-express this optimization, not directly in terms of the cluster labels $\mathbf{y}$, but instead in terms of the label outer product matrix $M = \mathbf{y}\mathbf{y}^\top$. The main advantage of doing so is that the inverse soft margin γ^{*-2} is in fact a convex function of M

$$\gamma^{*-2}(M) \ = \ \max_{\boldsymbol{\lambda}} \ \boldsymbol{\lambda}^\top \mathbf{e} - \frac{1}{2\beta}\langle K \circ \boldsymbol{\lambda}\boldsymbol{\lambda}^\top, M \rangle \quad \text{subject to} \quad 0 \le \boldsymbol{\lambda} \le 1 \tag{3.2}$$

The convexity of γ^{*-2} with respect to M is easy to establish since this quantity is just a maximum over linear functions of M (Boyd and Vandenberghe, 2004). This observation parallels one of the key insights of (Lanckriet et al., 2004), here applied to M instead of K.

Unfortunately, even though we can pose a convex objective, it does not allow us to immediately solve our problem because we still have to relate M to $\mathbf{y}$, and $M = \mathbf{y}\mathbf{y}^\top$ is not a convex constraint. Thus, the main challenge is to find a way to constrain M to ensure $M = \mathbf{y}\mathbf{y}^\top$ while respecting the class balance constraints $-\epsilon \leq \mathbf{e}^\top\mathbf{y} \leq \epsilon$. One obvious way to enforce $M = \mathbf{y}\mathbf{y}^\top$ would be to impose the constraint that $rank(M) = 1$, since combined with $M \in \{-1, +1\}^{n\times n}$ this forces M to have a decomposition $\mathbf{y}\mathbf{y}^\top$ for some $\mathbf{y} \in \{-1, +1\}^n$. Unfortunately, $rank(M) = 1$ is not a convex constraint on M (Helmberg, 2000).

The second key idea is to realize that one can *indirectly* enforce the desired relationship $M = \mathbf{y}\mathbf{y}^\top$ by imposing a different set of constraints on M. To do so, notice that any such M must encode an *equivalence relation* over the training points. That is, if $M = \mathbf{y}\mathbf{y}^\top$ for some $\mathbf{y} \in \{-1, +1\}^n$ then we must have

$$M_{ij} = \begin{cases} 1 & \text{if } y^i = y^j \\ -1 & \text{if } y^i \neq y^j \end{cases}$$

The properties that encode an equivalence relation are (1) reflexivity, $M_{ii} = 1$, (2) symmetry, $M_{ij} = M_{ji}$ and (3) transitivity, $M_{ij} = 1$ and $M_{jk} = 1$ implies $M_{ik} = 1$. For discrete $\{-1, +1\}$-valued matrices, these constraints can be imposed with linear inequalities. Unfortunately, transitivity require a cubic number of constraints to be imposed, which is impractical to store for most real problems. Fortunately, there is a nice result in the SDP literature that allows one to enforce all of these constraints in a quadratic SDP representation.

Lemma 2 *(Helmberg, 2000; Laurent and Poljak, 1995) Let* $M \succeq 0$ *and* $\operatorname{diag}(M) = \mathbf{e}$*, then* $-1 \leq M \leq 1$*. If in addition* $M \in \{-1, +1\}^{n\times n}$ *then* $M = \mathbf{y}\mathbf{y}^\top$ *for some* $\mathbf{y} \in \{-1, +1\}^n$*.*

Therefore to enforce the constraint $M = \mathbf{y}\mathbf{y}^\top$ for $\mathbf{y} \in \{-1, +1\}^n$ it suffices to impose the set of constraints: $M \in \{-1, +1\}^{n\times n}$, $M \succeq 0$ and $\operatorname{diag}(M) = \mathbf{e}$. Moreover, we can enforce the class balance constraint $-\epsilon \leq \mathbf{e}^\top\mathbf{y} \leq \epsilon$ by imposing the additional set of linear constraints: $-\epsilon\mathbf{e} \leq M\mathbf{e} \leq \epsilon\mathbf{e}$.

The combination of these two steps leads to our first main result: One can solve for a hard clustering $\mathbf{y}$ that maximizes the soft margin by solving a convex integer program. To accomplish this, one first solves for the equivalence relation matrix M in

$$\begin{aligned}&\min_{M\in\{-1,+1\}^{n\times n}}\ \max_{\boldsymbol{\lambda}}\ \boldsymbol{\lambda}^\top\mathbf{e}-\frac{1}{2\beta}\langle K\circ\boldsymbol{\lambda}\boldsymbol{\lambda}^\top, M\rangle\\ &\qquad\text{subject to}\quad 0\le\boldsymbol{\lambda}\le 1,\ \ \text{diag}(M)=\mathbf{e},\ \ M\succeq 0,\ \ -\epsilon\mathbf{e}\le M\mathbf{e}\le\epsilon\mathbf{e}\end{aligned}\tag{3.3}$$

Then, from the solution M^* the optimal cluster assignment $\mathbf{y}^*$ is recovered simply by setting $\mathbf{y}^*$ to any column vector in M^*.

Unfortunately, the formulation (3.3) is still not practical because nonlinear integer programming is still a hard computational problem. Therefore, we are compelled to take one further step and relax the integer constraint on M to obtain a convex optimization problem over a continuous parameter space

$$\begin{aligned}&\min_{M\in[-1,+1]^{n\times n}}\ \max_{\boldsymbol{\lambda}}\ \boldsymbol{\lambda}^\top\mathbf{e}-\frac{1}{2\beta}\langle K\circ\boldsymbol{\lambda}\boldsymbol{\lambda}^\top, M\rangle\\ &\qquad\text{subject to}\quad 0\le\boldsymbol{\lambda}\le 1,\ \ \text{diag}(M)=\mathbf{e},\ \ M\succeq 0,\ \ -\epsilon\mathbf{e}\le M\mathbf{e}\le\epsilon\mathbf{e}\end{aligned}\tag{3.4}$$

A more convenient form of this optimization problem is then given by the following theorem.

Theorem 1 *Solving (3.4) is equivalent to solving the semidefinite program*

$$\begin{aligned}\min_{M,\zeta,\boldsymbol{\mu},\boldsymbol{\nu}}\ \zeta\quad \textit{subject to}\quad & \textit{diag}(M)=\mathbf{e}, M\succeq 0, -\epsilon\mathbf{e}\le M\mathbf{e}\le\epsilon\mathbf{e}\\ & \boldsymbol{\mu}\ge 0, \boldsymbol{\nu}\ge 0\\ & \begin{bmatrix} M\circ K & \mathbf{e}+\boldsymbol{\mu}-\boldsymbol{\nu}\\ (\mathbf{e}+\boldsymbol{\mu}-\boldsymbol{\nu})^\top & \frac{2}{\beta}(\zeta-\boldsymbol{\nu}^\top\mathbf{e})\end{bmatrix}\succeq 0\end{aligned}\tag{3.5}$$

Proof It is easy to see that the convex optimization problem (3.4) is equivalent to

$$\begin{aligned}&\min_M\max_{\boldsymbol{\lambda}}\ \boldsymbol{\lambda}^\top\mathbf{e}-\frac{1}{2\beta}\boldsymbol{\lambda}^\top(K\circ M)\boldsymbol{\lambda}\\ &\quad\text{subject to}\quad 0\le\boldsymbol{\lambda}\le 1,\ \ -1\le M\le 1,\ \ \text{diag}(M)=\mathbf{e},\ \ M\succeq 0,\ \ -\epsilon\mathbf{e}\le M\mathbf{e}\le\epsilon\mathbf{e}\end{aligned}$$

which can be rewritten as

$$\begin{aligned}\min\nolimits_{M,\zeta}\ \zeta\quad\text{subject to}\quad & \zeta\ge\max\nolimits_{\boldsymbol{\lambda}}\boldsymbol{\lambda}^\top\mathbf{e}-\tfrac{1}{2\beta}\boldsymbol{\lambda}^\top(K\circ M)\boldsymbol{\lambda}\\ & 0\le\boldsymbol{\lambda}\le 1,\ \ \text{diag}(M)=\mathbf{e},\ \ M\succeq 0, -\epsilon\mathbf{e}\le M\mathbf{e}\le\epsilon\mathbf{e}\end{aligned}$$

Note that based on Lemma 2, the constraint $-1 \le M \le 1$ is implied by $\mathrm{diag}(M) = \mathbf{e}, M \succeq 0$ and thus can be removed.

Now let us look at the maximization problem

$$\omega(M) = \max_{\boldsymbol{\lambda}} \ \boldsymbol{\lambda}^\top \mathbf{e} - \frac{1}{2\beta}\boldsymbol{\lambda}^\top (K \circ M)\boldsymbol{\lambda} \quad \text{subject to } 0 \le \boldsymbol{\lambda} \le 1 \tag{3.6}$$

the Lagrangian of which is

$$\mathcal{L}(\boldsymbol{\lambda}, \boldsymbol{\mu}, \boldsymbol{\nu}) = \boldsymbol{\lambda}^\top \mathbf{e} - \frac{1}{2\beta}\boldsymbol{\lambda}^\top (K \circ M)\boldsymbol{\lambda} + \boldsymbol{\mu}^\top \boldsymbol{\lambda} + \boldsymbol{\nu}^\top(\mathbf{e} - \boldsymbol{\lambda})$$

By strong duality (Boyd and Vandenberghe, 2004), we have

$$\omega(M) = \max_{\boldsymbol{\lambda}} \ \min_{\boldsymbol{\mu}\ge 0, \boldsymbol{\nu} \ge 0} \ \mathcal{L}(\boldsymbol{\lambda}, \boldsymbol{\mu}, \boldsymbol{\nu}) = \min_{\boldsymbol{\mu}\ge 0, \boldsymbol{\nu} \ge 0} \ \max_{\boldsymbol{\lambda}} \ \mathcal{L}(\boldsymbol{\lambda}, \boldsymbol{\mu}, \boldsymbol{\nu})$$

By maximizing the Lagrangian over $\boldsymbol{\mu}$ and $\boldsymbol{\nu}$, at the optimum we have[1]

$$\boldsymbol{\lambda} = \beta (K \circ M)^{-1}(\mathbf{e} + \boldsymbol{\mu} - \boldsymbol{\nu})$$

and therefore can eliminate the primal variable in the Lagrangian, to obtain the dual of (3.6)

$$\omega(M) = \min_{\boldsymbol{\mu}\ge 0, \boldsymbol{\nu} \ge 0} \ \frac{\beta}{2}(\mathbf{e} + \boldsymbol{\mu} - \boldsymbol{\nu})^\top (K \circ M)^{-1}(\mathbf{e} + \boldsymbol{\mu} - \boldsymbol{\nu}) + \boldsymbol{\nu}^\top \mathbf{e}$$

Now the constraint $\zeta \ge \omega(M)$ can be satisfied if and only if for any ζ, there exist $\boldsymbol{\mu} \ge 0, \boldsymbol{\nu} \ge 0$ such that

$$\zeta \ge \frac{\beta}{2}(\mathbf{e} + \boldsymbol{\mu} - \boldsymbol{\nu})^\top (K \circ M)^{-1}(\mathbf{e} + \boldsymbol{\mu} - \boldsymbol{\nu}) + \boldsymbol{\nu}^\top \mathbf{e}$$

or

$$\frac{2}{\beta}(\zeta - \boldsymbol{\nu}^\top \mathbf{e}) \ge (\mathbf{e} + \boldsymbol{\mu} - \boldsymbol{\nu})^\top (K \circ M)^{-1}(\mathbf{e} + \boldsymbol{\mu} - \boldsymbol{\nu}) \tag{3.7}$$

[1]We assume that $(K \circ M)^{-1}$ exists. Otherwise, the pseudo inverse $(K \circ M)^{+}$ can be used instead and the rest of the derivation still holds (Bie and Cristianini, 2003); or one can add a small value to the diagonal of $K \circ M$ to make it non-singular.

Using the Schur complement lemma (Horn and Johnson, 1985),[2] the constraint (3.7) can now be reformulated as

$$\begin{bmatrix} M \circ K & \mathbf{e}+\boldsymbol{\mu}-\boldsymbol{\nu} \\ (\mathbf{e}+\boldsymbol{\mu}-\boldsymbol{\nu})^\top & \frac{2}{\beta}(\zeta - \boldsymbol{\nu}^\top \mathbf{e}) \end{bmatrix} \succeq 0$$

Consequently, the problem (3.4) is turned into

$$\min_{M,\zeta,\boldsymbol{\mu},\boldsymbol{\nu}} \zeta \quad \text{subject to} \quad \begin{aligned} & diag(M) = \mathbf{e}, M \succeq 0, -\epsilon\mathbf{e} \le M\mathbf{e} \le \epsilon\mathbf{e} \\ & \boldsymbol{\mu} \ge 0, \boldsymbol{\nu} \ge 0 \\ & \begin{bmatrix} M \circ K & \mathbf{e}+\boldsymbol{\mu}-\boldsymbol{\nu} \\ (\mathbf{e}+\boldsymbol{\mu}-\boldsymbol{\nu})^\top & \frac{2}{\beta}(\zeta - \boldsymbol{\nu}^\top \mathbf{e}) \end{bmatrix} \succeq 0 \end{aligned}$$

which is a semidefinite program. ▮

This gives us the second main result: To solve for a soft clustering $\mathbf{y}$ that approximately maximizes the soft margin, first solve the semidefinite program (3.5), and then from the solution matrix M^* recover the soft cluster assignment $\mathbf{y}$ by setting $\mathbf{y} = \sqrt{\alpha_1}\mathbf{v}_1$, where $\alpha_1, \mathbf{v}_1$ are the maximum eigenvalue and corresponding eigenvector of M^*.[3]

3.3 Comparison to Existing Unsupervised Methods

As mentioned previously, a large number of kernel based clustering algorithms have been proposed in the literature, most of which follow the "spectral clustering" approach. An

[2]**Schur Complement Lemma:** Given a symmetric matrix

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

where $A \succ 0, C \succeq 0$, we define the Schur complement of A in X by $S = C - B^\top A^{-1} B$. Then

$$\begin{bmatrix} A & B \\ B^\top & C \end{bmatrix} \succ 0 \iff S \succ 0 \qquad \text{and} \qquad \begin{bmatrix} A & B \\ B^\top & C \end{bmatrix} \succeq 0 \iff S \succeq 0$$

[3]One could also employ randomized rounding to choose a hard class assignment $\mathbf{y}$.

interesting question is how the standard graph cut criterion used in spectral clustering relates to the large margin principle I am exploring in this thesis. A recent paper (Rahimi and Recht, 2004) sheds some light on this relationship.

In (Rahimi and Recht, 2004), the authors analyze a special case, previously introduced in Section 2.2.1, where the kernel matrix K is both positive semidefinite and nonnegative. In their analysis they assume furthermore that $diag(K) = \mathbf{e}$, which, for example, holds for the standard RBF kernel (2.20). Given these assumptions, one can see that all training points $\mathbf{x}^1, ..., \mathbf{x}^n$ are mapped onto the surface of the unit sphere in the positive orthant, since $\|\boldsymbol{\phi}(\mathbf{x}^i)\| = \sqrt{K_{ii}} = 1$ and $\boldsymbol{\phi}(\mathbf{x}^i) \cdot \boldsymbol{\phi}(\mathbf{x}^j) = K_{ij} \geq 0$. In this intuitive scenario, Rahimi and Recht (2004) observe that spectral clustering, and in particular, normalized cut, is equivalent to partitioning the data with a hyperplane that maximizes a weighted average distance of the n data points to the hyperplane:

$$\max \frac{1}{n} \sum_{i=1}^{n} \frac{(dist_i)^2}{\cos\theta_i} \tag{3.8}$$

where $dist_i$ is the Euclidean distance between data point i and the hyperplane (defined in (2.2)), and θ_i is the angle between the i-th data point with the mean of the data in the feature space. Thus the criterion prefers hyperplanes that obtain a large distance to points that are nearly orthogonal to the data mean, where $1/\cos\theta_i$ serves as the weight of data point i.

By contrast, the principle of maximum margin clustering is to find a hyperplane that maximizes the distance of the *nearest* data point to the hyperplane. More precisely, I use a soft margin formulation (subject to a relaxation) to find a hyperplane that minimizes the hinge loss (2.7). Intuitively, the hinge loss criterion is more robust against outliers and nonlinear class structure away from the boundaries, whereas the normalized cut criterion remains sensitive to these. For example, consider the effect of an outlier. Under (3.8) the outlier's contribution to the loss will be proportional to its squared distance to the hyperplane, times $1/\cos\theta_i$, which must grow to ∞ as the outlier moves further away from the mean (i.e., θ_i goes to zero). By contrast, an outlier's contribution to the hinge loss (2.7) is only given by its absolute distance to the hyperplane. Below we find that the outcome usually works in favor of the maximum margin approach: a sufficiently rich feature space defined by a kernel usually allows reasonable separation by a hyperplane, whereas nonlinearities in the within class structure appears to weaken the performance of normalized cut.

One important note is that, currently, the computational cost of minimizing the two objectives is not the same. As shown in Section 2.2.3, a relaxed form of the normalized cut criterion can be minimized by solving a generalized smallest eigenvalue problem. However, the algorithm I have developed for minimizing a relaxed form of the maximum margin criterion involves solving a semidefinite program, and hence is much more expensive in practice (although still polynomial time (Nesterov and Nemirovskii, 1994)).

3.4 Experimental Results: Unsupervised Training

I implemented the maximum margin clustering algorithm based on the semidefinite programming formulation (3.5), using the SeDuMi library (Sturm, 1999), and ran various experiments to verify its intuitive behavior and also to determine its potential effectiveness on real data sets.

First, to illustrate the difference of k-means clustering, spectral clustering and maximum margin clustering, I ran a simple synthetic experiment using data on the unit sphere, as discussed above. Here the number of data points is small enough to run k-means to optimality, we can see that due to the existence of the outlier, k-means separates the rest of the data points with the outlier to minimize the within class distance. On the other hand, normalized cut can be thought of as partitioning the data with a hyperplane that maximizes a weighted average of the squared distances to the hyperplane in this case. Because normalized cut puts more weight on data points that are nearly orthogonal to the mean of data (i.e., data points that are far from the class mean), this criterion prefers a hyperplane that is is attracted to these data points; thus inducing outlier sensitivity. As shown in Figure 3.1, one can see by comparing the left and right column that when the outlier moves further from the data mean the solution of normalized cut is shifted, while the maximum margin solution remains unaffected. Moreover, it can be observed that in the third row, the semidefinite approach returns with the optimal solution to the maximum margin clustering problem, therefore, this comparison shown in Figure 3.1 demonstrates not only the difference of the algorithms themselves, but also the difference of the underlying clustering principles.

Note that this particular experiment was only designed to illustrate the basic difference between the clustering algorithms discussed above. Here I only used a linear kernel to illustrate the point. It is possible that spectral clustering and k-means can avoid the specific problem shown above by using an appropriate kernel, but that is missing the point that they remain sensitive to outliers in whatever feature space is implied by the kernel.

In the next set of experiments I compared the performance of the maximum margin clustering technique to the spectral clustering method of (Ng et al., 2001) as well as straightforward k-means clustering on a wider range of data sets. Both maximum margin clustering and spectral clustering were run with the same radial basis function kernel and matching width parameters. In fact, in each case, I chose the best width parameter *for spectral clustering* by searching over a small set of five widths related to the scale of the problem. In addition, the slack parameter for maximum margin clustering was simply set to an arbitrary value.[4]

To assess clustering performance I used mis-assignment rate introduced in Section 2.2.4 as the performance measure. That is, I take a set of *labeled* data, remove the labels, run the clustering algorithms, find the optimal correspondence between the true class labels and the resulting clusters, and then measure the number of misclassifications.

The first experiments were conducted on the synthetic data sets depicted in Figure 3.2. Table 3.1 shows that for the first three sets of data (Gaussians, Circles, AI) maximum margin and spectral clustering obtained identical small error rates, which were in turn significantly smaller than those obtained by k-means. However, maximum margin clustering demonstrates a substantial advantage on the fourth data set (Joined Circles) over both spectral and k-means clustering.

I also conducted clustering experiments on the real data sets, two of which are depicted in Figures 3.3 and 3.4: a database of 100 images of handwritten digits of twos and threes (Figure 3.3), and a database of 76 face images of two people (Figure 3.4). The last two columns of Table 3.1 show that maximum margin clustering obtains a slight advantage on

[4]It turns out that the slack parameter β did not have a significant effect in these experiments, so I just set it to $\beta = 0.01$ for all of the experiments reported here.

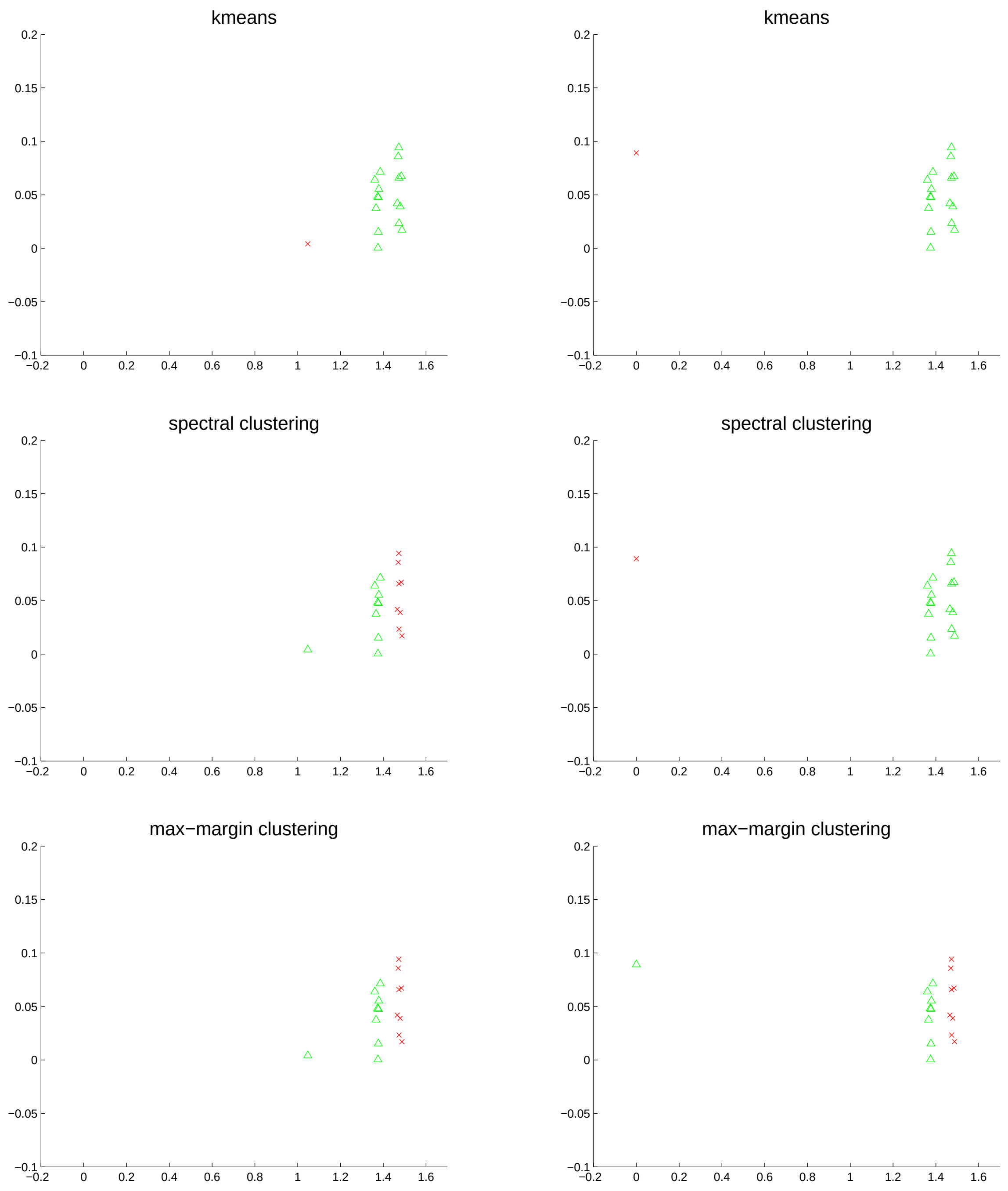


Figure 3.1: Comparison of k-means, normalized cut and maximum margin clustering when outliers are present. The three rows show the results produced by k-means, normalized cut and maximum margin clustering respectively. The two axes show angular positions on the unit sphere in spherical (ϕ, θ) coordinates.

Clustering:	**Semidefinite**	Spectral	Kmeans
Gaussians	**1.2**	1.2	5
Circles	**0**	0	50.0
AI	**0**	0	38.5
Joined Circles	**1.0**	24.0	50.0
Digits	**3.0**	6.0	7.0
Faces	**0**	16.7	24.4

Table 3.1: Percentage misclassification errors of the various clustering algorithms on the various data sets.

the handwritten digits data, and a significant advantage on the faces data.

The running time of the maximum margin clustering algorithm on 50 data points varies from 10 to 30 seconds on a 2.4 GHz AMD Opteron 250 processor, depending on different problems, however it can increase up to around 10 minutes on data sets with 100 examples.

Although these results are limited, they demonstrate that the maximum margin principle offers a potentially very effective approach to kernel based clustering. In particular, maximum margin clustering with RBF kernels appears to be very effective at separating classes, even when data has a complex distribution, and moreover seems to demonstrate a greater robustness to outliers on the training data than the standard spectral clustering method.

One nice advantage of the maximum margin approach developed in this chapter is that, in principle, it is out of sample. That is, once the clustering has been determined, (2.21) can be used to recover $\boldsymbol{\lambda}$. This defines a universal classification rule (2.22) that can be applied to any future data without running the training algorithm again.

3.5 Semi-supervised Two-class SVMs

In addition to clustering, I have the further goal of adapting the maximum margin approach to *semi-supervised* learning. For semi-supervised learning, we assume we are given a labeled training set $(\mathbf{x}^1, y^1), ..., (\mathbf{x}^t, y^t)$ as well as an unlabeled training set $\mathbf{x}^{t+1}, ..., \mathbf{x}^n$, and the goal

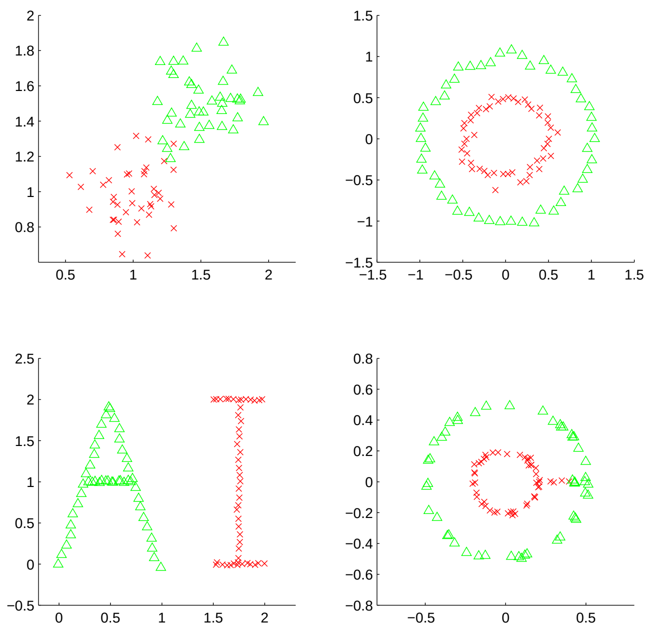


Figure 3.2: Four artificial data sets used in the binary clustering experiments.

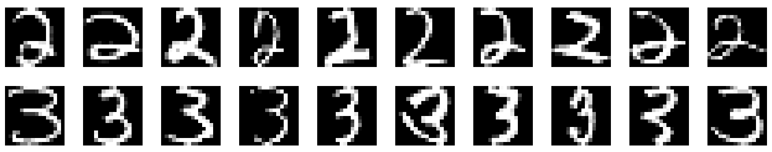


Figure 3.3: A sampling of the handwritten digits (twos and threes).

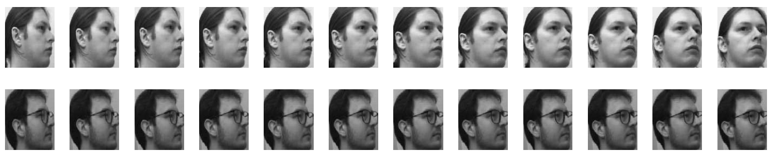


Figure 3.4: A sampling of the face data (two people).

is to combine the information in these two data sets to produce a more accurate classifier.

In the context of large margin classifiers, many techniques have been proposed for incorporating unlabeled data in an SVM, most of which are intuitively based on ensuring that large margins are also preserved on the unlabeled training data (Joachims, 1999; Bennett and Demiriz, 1998), just as in our case. However, none of these previous proposals have formulated a convex optimization objective that is guaranteed to yield a globally optimal approximation, as I propose in Section 3.1.

For our procedure, extending the maximum margin clustering approach of Section 3.1 to semi-supervised training is easy: We simply add constraints on the matrix M to force it to respect the observed equivalence relations among the labeled training data. In addition, we impose the constraint that each unlabeled example belongs to the same class as at least one labeled training example. These conditions can be enforced with the simple set of additional linear constraints

$$\begin{array}{ll} \mathcal{S}_1: & M_{ij} = y^i y^j \quad \text{for } \textit{labeled} \text{ examples } i, j \in \{1, ..., t\} \\ \mathcal{S}_2: & \sum_{i=1}^{t} M_{ij} \geq 2 - t \quad \text{for } \textit{unlabeled} \text{ examples } j \in \{t+1, ..., n\} \end{array}$$

Note that the observed training labels y^i for $i \in \{1, ..., t\}$ are *constants*, and therefore the new constraints are still linear in the parameters of M that are being optimized.

The resulting training procedure is similar to that of (Bie and Cristianini, 2003), with the addition of the class balance constraint $-\epsilon \mathbf{e} \leq M\mathbf{e} \leq \epsilon \mathbf{e}$ and $\mathcal{S}_2$ which enforce two classes and facilitate the ability to perform clustering on the unlabeled examples.

3.6 Comparison to Existing Semi-supervised Methods

The most well known technique for semi-supervised training with SVMs is the transductive SVM algorithm proposed in (Joachims, 1999), which is also based on large margin criterion. As in my work, given a labeled training set $(\mathbf{x}^1, y^1), ..., (\mathbf{x}^t, y^t)$ and an unlabeled training set $\mathbf{x}^{t+1}, ..., \mathbf{x}^n$, the goal of transductive SVMs is to find a hyperplane $(\mathbf{w}, b)$ and a labelling $(\hat{y}^{t+1}, ..., \hat{y}^n)$ of the unlabeled part such that the hyperplane separates the labeled data and unlabeled data with the largest margin. This is formulated as

$$\begin{aligned}\min_{\hat{y}^{t+1},\ldots,\hat{y}^n,\mathbf{w},b,\xi_1,\ldots\xi_n} \quad & \frac{1}{2}\|\mathbf{w}\|^2 + \frac{1}{\beta}\sum_{i=1}^{t}\xi_i + \frac{1}{\beta'}\sum_{i=t+1}^{n}\xi_i \\ \text{subject to} \quad & y^i(\mathbf{w}\cdot\mathbf{x}^i + b) \geq 1 - \xi_i \ \ \forall_{i=1}^{t} \\ & \hat{y}^i(\mathbf{w}\cdot\mathbf{x}^i + b) \geq 1 - \xi_i \ \ \forall_{i=t+1}^{n} \\ & \xi_i \geq 0 \ \ \forall i\end{aligned} \tag{3.9}$$

where β and β' are the slack parameters.

A naive approach to solve (3.9) is to enumerate all possible configurations of $(\hat{y}^{t+1}, \ldots, \hat{y}^n)$, however, this is impractical. Therefore (Joachims, 1999) proposes an iterative algorithm instead. In each iteration two examples are identified if changing their labels could decrease the objective function, and then the labels of these examples are switched. This algorithm will converge with a finite number of iterations, however, global solution is not guaranteed. By contrast, the algorithm I proposed in Section 3.5 solves a convex optimization problem, and hence obtains a global solution. Below we find that the globally optimized outcome tends to produce more accurate results than (3.9).

3.7 Experimental Results: Semi-supervised Training

I tested my approach to semisupervised learning on various two class data sets from the UCI repository. For these UCI data sets, I took 50 random examples from both classes. In each case, I evaluated the techniques transductively. That is, I randomly subsampled 1/10 of the data as the labeled part and the rest as the unlabeled part, held out the labels of the unlabeled portion, trained the semi-supervised techniques, reclassified the unlabeled examples using the learned results, and measured the misclassification error on the held out labels. I compared the performance of the proposed technique to the transductive SVM technique of (Joachims, 1999) and standard SVM. Linear kernel was used here, and β parameter was set to 0.01 for the maximum margin approach, while the best β value was chosen for transductive SVM and standard SVM respectively. The results were averaged over 5 times.

Semi-sup:	**Semidefinite**	TSVM	Spectral	supSVM
Digits 1-7	**3.3** $\pm$**0**	4.6 $\pm$0.9	4.2 $\pm$0.2	5.1 $\pm$1.3
Digits 2-3	**4.7** $\pm$**0.2**	5.4 $\pm$2.7	6.4 $\pm$0.2	10.9 $\pm$4.9
UCI Australian	**32.0** $\pm$**1.2**	38.7 $\pm$3.5	48.7 $\pm$7.4	42.2 $\pm$3.6
UCI Flare	**34.0** $\pm$**1.0**	33.3 $\pm$1.4	40.7 $\pm$7.3	35.6 $\pm$2.6
UCI Vote	**14.0** $\pm$**0.7**	17.5 $\pm$2.9	13.8 $\pm$0.4	19.3 $\pm$6.6

Table 3.2: Percentage misclassification errors of the various semisupervised learning algorithms on the various data sets ($\pm$ one standard error of the mean). TSVM is due to (Joachims, 1999). Note that supSVM uses no unlabeled data.

Here one can see that the maximum margin approach based on semidefinite programming generally outperforms the approach of (Joachims, 1999). Table 3.2 shows that the maximum margin method is effective at exploiting unlabeled data to improve the prediction of held out labels. In every case, it significantly reduces the error of plain SVM, and obtains the best overall performance of the semisupervised learning techniques we have investigated.

3.8 Conclusion

In this chapter I have proposed a simple principle for two-class clustering based on the maximum margin principle popularized by supervised SVMs: we seek a labelling that would yield an optimal SVM classifier on the training data. Interestingly, this criterion can be approximately optimized using an efficient semidefinite programming formulation. Moreover, semi-supervised learning can also be easily achieved in a unified framework. The results on both clustering and semi-supervised learning are competitive with, and sometimes exceed the state of the art. Overall, margin maximization appears to be an effective way to achieve a unified approach to these different two-class learning problems. Importantly, the principle proposed here, together with the advances of margin maximization approaches beyond the simple two-class case, facilitates us to extend the framework in anther direction to the multi-class and even multivariate settings.

Chapter 4

Unsupervised Multi-class Support Vector Machines

In many real-world applications, learning problems involve more than two classes. There are many ways to extend two-class classification schemes to multi-class classification schemes as discussed in Section 2.1.2, including one versus the rest classification, all pairs classification, error-correcting output coding, and hierarchical classification. However, from an SVM perspective the most elegant approach to handling multi-class problems is to directly minimize the multi-class margin loss (2.27) introduced by (Crammer and Singer, 2001), as discussed in Section 2.1.2.

In this chapter, I extend the previous two-class methods to general multi-class SVM training algorithms for unsupervised and semi-supervised learning. Essentially, the methodology can now be broken down into a series of steps: First, one treats the missing classification labels as variables to be optimized jointly with the SVM parameters. Second, one formulates the dual form of the supervised maximum margin problem. This formulation will be expressed in terms of pairwise comparisons of the supervised y-labels. Third, one then re-expresses the problem in terms of indicator variables that compare y-labels, as opposed to the y-labels themselves. The optimization criterion will then become a convex function in the indicator variables, even when it was generally not a convex function of the original y-labels. Finally, one relaxes the indicator variables to real values, and possibly relaxes additional indicator constraints, to obtain a convex optimization problem.

This problem can either be expressed as a pure semidefinite program, or a more compact semidefinite program with a constraint generation phase that calls the original maximum margin algorithm (a quadratic program) with the current soft indicators.

Even though the strategy employed for the multi-class case is similar to that employed above, there are some significant complications to deriving an effective training procedure.

4.1 Unsupervised Multi-class SVMs

As before, I start with the supervised case. Assume we are given labeled training examples $(\mathbf{x}^1, y^1), ..., (\mathbf{x}^n, y^n)$ where each example is assigned a label from a fixed finite set $y^i \in \{1, ..., \kappa\}$. As shown in (2.27), the multi-class training problem can be reformulated as a quadratic program, which I repeat here for convenience

$$\begin{aligned} \omega(\mathbf{y}) \quad = \quad & \min_{\mathbf{w},\boldsymbol{\xi}} \frac{1}{2}\beta\|\mathbf{w}\|^2 + \sum_{i=1}^{n} \xi_i \\ & \text{subject to} \;\; \mathbf{w}\cdot(\boldsymbol{\phi}(\mathbf{x}^i, y^i) - \boldsymbol{\phi}(\mathbf{x}^i, r)) + \delta_{y^i,r} \geq 1 - \xi_i \quad \forall i, r \\ & \qquad\qquad\quad \xi_i \geq 0 \quad \forall i \end{aligned} \tag{4.1}$$

Here ω is the multi-class analog of the inverse squared margin γ^{*-2}, and depends on the given multi-class labels $\mathbf{y}$. My main result depends crucially on being able to reformulate the quadratic program (4.1) so that it can be expressed entirely in terms of equivalence relation matrices instead of individual y-labels. To achieve this, I need to derive a different formulation of the dual from that given in (2.28) (Crammer and Singer, 2001).

With some work, one can show that the following quadratic program is equivalent to the dual of (4.1).

Theorem 2 *The dual quadratic program of (4.1) can be reformulated as*

$$\omega(M, D) = \max_{\boldsymbol{\Lambda}} \; Q(\Lambda, M, D) \;\; \textit{subject to} \; \Lambda \geq 0, \; \Lambda\mathbf{e} = \mathbf{e}$$

$$\begin{aligned} \textit{where} \quad Q(\Lambda, M, D) = \; & n - \langle D, \Lambda\rangle - \tfrac{1}{2\beta}\langle K, M\rangle + \\ & \tfrac{1}{\beta}\langle KD, \Lambda\rangle - \tfrac{1}{2\beta}\langle \Lambda\Lambda^\top, K\rangle \end{aligned} \tag{4.2}$$

Here D and Λ are $n \times \kappa$ matrices, M and K are $n \times n$ matrices, and we define the equivalence relation indicator matrices M and D such that $M_{ij} = \delta_{y^i, y^j}$ and $D_{ir} = \delta_{y^i, r}$ respectively.

Note: An important consequence of the definitions above, which I exploit below, is that $M = DD^\top$.

Proof Given the primal multi-class training problem

$$\begin{aligned} \omega &= \min_{\mathbf{w}, \boldsymbol{\xi}} \frac{\beta}{2}\|\mathbf{w}\|^2 + \sum_{i=1}^{n} \xi_i \\ & \quad \text{subject to } \mathbf{w} \cdot \left(\boldsymbol{\phi}(\mathbf{x}^i, y^i) - \boldsymbol{\phi}(\mathbf{x}^i, r)\right) \geq 1 - \delta_{y^i, r} - \xi_i \quad \forall i, r \end{aligned} \tag{4.3}$$

where

$$\boldsymbol{\phi}(\mathbf{x}, y) \;=\; \begin{bmatrix} \boldsymbol{\phi}(\mathbf{x}) \cdot \delta_{y,1} \\ \ldots \\ \boldsymbol{\phi}(\mathbf{x}) \cdot \delta_{y,\kappa} \end{bmatrix},$$

we can get the Lagrangian of this optimization problem by adding a set of dual variables Λ

$$\begin{aligned} L = \; & \frac{\beta}{2}\|\mathbf{w}\|^2 + \sum_{i=1}^{n} \xi_i - \sum_{i=1}^{n} \sum_{r=1}^{\kappa} \Lambda_{ir} \left(\mathbf{w} \cdot \left(\boldsymbol{\phi}(\mathbf{x}^i, y^i) - \boldsymbol{\phi}(\mathbf{x}^i, r)\right) - (1 - \delta_{y^i, r} - \xi_i)\right) \\ & \Lambda \geq 0 \end{aligned} \tag{4.4}$$

where Λ is an $n \times \kappa$ matrix.

By minimizing over $\mathbf{w}, \boldsymbol{\xi}$, we can obtain

$$\mathbf{w} = \frac{1}{\beta} \sum_{i,r} \Lambda_{ir} (\boldsymbol{\phi}(\mathbf{x}^i, y^i) - \boldsymbol{\phi}(\mathbf{x}^i, r)) \tag{4.5}$$

$$\sum_{r} \Lambda_{ir} = 1 \quad \forall i \tag{4.6}$$

Plugging (4.5) and (4.6) into (4.4), and adding the constraints that $\Lambda \geq 0$, we obtain the dual optimization problem

$$\begin{aligned} & \max_{\boldsymbol{\Lambda}} \sum_{ir} \Lambda_{ir} (1 - \delta_{y^i, r}) - \frac{1}{2\beta} \| \sum_{ir} \Lambda_{ir} \left(\boldsymbol{\phi}(\mathbf{x}^i, y^i) - \boldsymbol{\phi}(\mathbf{x}^i, r)\right) \|^2 \\ & \qquad \text{subject to} \quad \sum_{r} \Lambda_{ir} = 1, \forall i \text{ and } \Lambda_{ir} \geq 0, \forall i, r \end{aligned}$$

Notice that $\boldsymbol{\phi}(\mathbf{x}, y) \cdot \boldsymbol{\phi}(\mathbf{x}', y') = \delta_{y,y'}\boldsymbol{\phi}(\mathbf{x}) \cdot \boldsymbol{\phi}(\mathbf{x}')$, therefore, the dual can be further reformulated as

$$\begin{aligned}
& \max\nolimits_\Lambda \quad Q(\Lambda, M, D) \\
= \; & \max\nolimits_\Lambda \quad n - \textstyle\sum_{ir} \Lambda_{ir}\delta_{y^i,r} - \frac{1}{2\beta}\sum_{ij}\sum_{rs} \Lambda_{ir}\Lambda_{js}\left(\boldsymbol{\phi}(\mathbf{x}^i) \cdot \boldsymbol{\phi}(\mathbf{x}^j)\right)\left(\delta_{y^i,y^j} - \delta_{y^i,s} - \delta_{y^j,r} + \delta_{r,s}\right) \\
= \; & \max\nolimits_\Lambda \quad n - \textstyle\sum_{ir} \Lambda_{ir}D_{ir} - \frac{1}{2\beta}\sum_{ij}\sum_{rs} \Lambda_{ir}\Lambda_{js}K_{ij}(M_{ij} - D_{is} - D_{jr} + \delta_{rs}) \\
= \; & \max\nolimits_\Lambda \quad n - \langle \Lambda, D\rangle - \tfrac{1}{2\beta}\langle K, M\rangle + \tfrac{1}{\beta}\langle K, D\Lambda^\top\rangle - \tfrac{1}{2\beta}\langle \Lambda\Lambda^\top, K\rangle \\
& \qquad\qquad \text{subject to} \quad \Lambda \geq 0, \; \Lambda\mathbf{e} = \mathbf{e}
\end{aligned}$$

where $M_{ij} = \delta_{y^i,y^j}$ and $D_{ir} = \delta_{y^i,r}$. ∎

This is not the same formulation of the dual to (4.1) given in (Crammer and Singer, 2001). Importantly, this reformulation of the dual expresses the problem explicitly in terms of the equivalence relations M and D—which gives us a necessary advantage over the formulation of the dual given in (Crammer and Singer, 2001).

The main result can now be established. We would like to compute a solution to the problem

$$\min_{M,D} \omega(M, D) \qquad \text{subject to} \quad M = DD^\top$$
$$M \in \{0,1\}^{n\times n}, \; D \in \{0,1\}^{n\times\kappa}$$

A solution to this problem will minimize the inverse square margin criterion ω and thus maximize the original margin criterion. However, just as in the two-class case, we need to worry about class balance, because a large margin value can always be achieved by eliminating classes. To impose class balance in the multi-class case we add the constraint $(\frac{1}{\kappa} - \epsilon)n\mathbf{e} \leq M\mathbf{e} \leq (\frac{1}{\kappa} + \epsilon)n\mathbf{e}$ for some ϵ.

The overall formulation, to this point, enjoys the advantage that the objective $\omega(M, D)$ is jointly convex in M and D. Unfortunately, neither the integer constraints nor the nonlinear constraint $M = DD^\top$ are convex. Therefore, as before, we need to derive a convex relaxation to this problem that preserves as much of the structure of the problem as possible. To do so, we first relax the constraint that M and D be $\{0,1\}$-valued, and instead allow them to take values in $[0, 1]$. However, we also have to cope with the nonlinear equality constraint $M = DD^\top$. To remove the non-convexity implicit in the nonlinear equality, we replace it with the convex inequalities $M \succeq DD^\top$ and $\mathrm{diag}(M) = \mathbf{e}$. The resulting problem is a convex optimization over M and D.

$$\begin{aligned} \min_{M,D}\ \omega(M, D)\ \text{ subject to} \quad & 0 \le M \le 1,\ \ 0 \le D \le 1 \\ & \mathrm{diag}(M) = \mathbf{e},\ \ M \succeq DD^\top \\ & (\tfrac{1}{\kappa} - \epsilon)n\mathbf{e} \le M\mathbf{e} \le (\tfrac{1}{\kappa} + \epsilon)n\mathbf{e} \end{aligned} \tag{4.7}$$

This in fact yields the convex optimization problem we wish to solve.

To tackle this problem in practice, it is convenient to reformulate it as a semidefinite program.

Theorem 3 *The convex optimization problem (4.7) is equivalent to*

$$\begin{aligned} & \min_{M,D,V,\boldsymbol{\alpha},\zeta} \zeta \qquad \textit{subject to} \\ & \begin{bmatrix} I \otimes K & \mathbf{p} \\ \mathbf{p}^\top & \frac{2}{\beta}\zeta + \frac{1}{\beta^2}\langle K, M\rangle + \frac{2}{\beta}\boldsymbol{\alpha}^\top \mathbf{e} \end{bmatrix} \succeq 0 \\ & \begin{bmatrix} I & D^\top \\ D & M \end{bmatrix} \succeq 0 \\ & \textit{diag}(M) = \mathbf{e},\ \ 0 \le M \le 1,\ \ 0 \le D \le 1,\ \ V \ge 0 \\ & (\tfrac{1}{\kappa} - \epsilon)n\mathbf{e} \le M\mathbf{e} \le (\tfrac{1}{\kappa} + \epsilon)n\mathbf{e} \end{aligned} \tag{4.8}$$

where $\mathbf{p} = \textit{vec}(\frac{1}{\beta}KD - D + V + \boldsymbol{\alpha}\mathbf{e}^\top)$.[1]

[1]The notation $\mathrm{vec}(X)$ means turning X into a vector by concatenating its columns. $A \otimes B$ denotes the Kronecker product (Horn and Johnson, 1985). If A is an $m \times n$ and B is a $p \times q$ matrix, then $A \otimes B$ is a block matrix of $mp \times nq$, and

$$A \otimes B = \begin{bmatrix} a_{11}B & \dots & a_{1n}B \\ \vdots & \ddots & \vdots \\ a_{m1}B & \dots & a_{mn}B \end{bmatrix}$$

Proof We have

$$\omega(M,D) = \max_{\Lambda} Q(\Lambda, M, D) \quad \text{subject to} \quad \Lambda \geq 0,\ \Lambda\mathbf{e} = \mathbf{e} \tag{4.9}$$

The corresponding Lagragian is

$$\begin{aligned}\mathcal{L}(\Lambda, V, \boldsymbol{\alpha}) &= -\tfrac{1}{2\beta}\langle \Lambda\Lambda^\top, K\rangle + \tfrac{1}{\beta}\langle KD, \Lambda\rangle - \langle \Lambda, D\rangle - \tfrac{1}{2\beta}\langle K, M\rangle \\ &\quad + \langle V, \Lambda\rangle + \langle \boldsymbol{\alpha}\mathbf{e}^\top, \Lambda\rangle - \boldsymbol{\alpha}^\top\mathbf{e} \\ &= -\tfrac{1}{2\beta} tr(K\Lambda\Lambda^\top) + tr((\tfrac{1}{\beta}D^\top K - D^\top + V^\top + \mathbf{e}\boldsymbol{\alpha}^\top)\Lambda) - \tfrac{1}{2\beta}\langle K, M\rangle - \boldsymbol{\alpha}^\top\mathbf{e} \\ &\quad \text{subject to} \qquad V \geq 0\end{aligned}$$

This can be further reformulated into

$$\mathcal{L}(\Lambda, V, \boldsymbol{\alpha}) = -\frac{1}{2\beta}\boldsymbol{\lambda}^\top(I \otimes K)\boldsymbol{\lambda} + \mathbf{p}^\top\boldsymbol{\lambda} - \frac{1}{2\beta}\langle K, M\rangle - \boldsymbol{\alpha}^\top\mathbf{e} \tag{4.10}$$

based on the rule $tr(AXBX^\top) = vec(X)^\top(B \otimes A)vec(X)$, where

$$\boldsymbol{\lambda} = vec(\Lambda), \quad \mathbf{p} = vec(\frac{1}{\beta}KD + V - D + \boldsymbol{\alpha}\mathbf{e}^\top)$$

By maximizing the Lagragian over $\boldsymbol{\lambda}$, at the optimum we have:[2]

$$\boldsymbol{\lambda} = \beta(I \otimes K)^{-1}\mathbf{p}$$

Therefore, the dual of (4.9) is:

$$\omega(M,D) = \min_{V,\boldsymbol{\alpha}} \frac{\beta}{2}\mathbf{p}^\top(I \otimes K)^{-1}\mathbf{p} - \frac{1}{2\beta}\langle K, M\rangle - \boldsymbol{\alpha}^\top\mathbf{e} \quad \text{subject to} \quad V \geq 0$$

This can be reformulated as an equivalent semidefinite program using similar derivation as in the proof of Theorem 1 , yielding

$$\min_{M,D,V,\boldsymbol{\alpha},\zeta} \zeta \qquad \text{subject to}$$

[2]Here we assume that $(I \otimes K)$ is invertible. If not, similar fixes to those discussed in the proof of Theorem 1 of Chapter 3 would apply equally here.

$$\begin{bmatrix} I \otimes K & \mathbf{p} \\ \mathbf{p}^\top & \frac{2}{\beta}\zeta + \frac{1}{\beta^2}\langle K, M\rangle + \frac{2}{\beta}\boldsymbol{\alpha}^\top \mathbf{e} \end{bmatrix} \succeq 0 \tag{4.11}$$

$$\begin{bmatrix} I & D^\top \\ D & M \end{bmatrix} \succeq 0$$

$$\mathrm{diag}(M) = \mathbf{e}, \;\; 0 \leq M \leq 1, \;\; 0 \leq D \leq 1, \;\; V \geq 0$$

$$(\tfrac{1}{\kappa} - \epsilon)n\mathbf{e} \leq M\mathbf{e} \leq (\tfrac{1}{\kappa} + \epsilon)n\mathbf{e}$$

∎

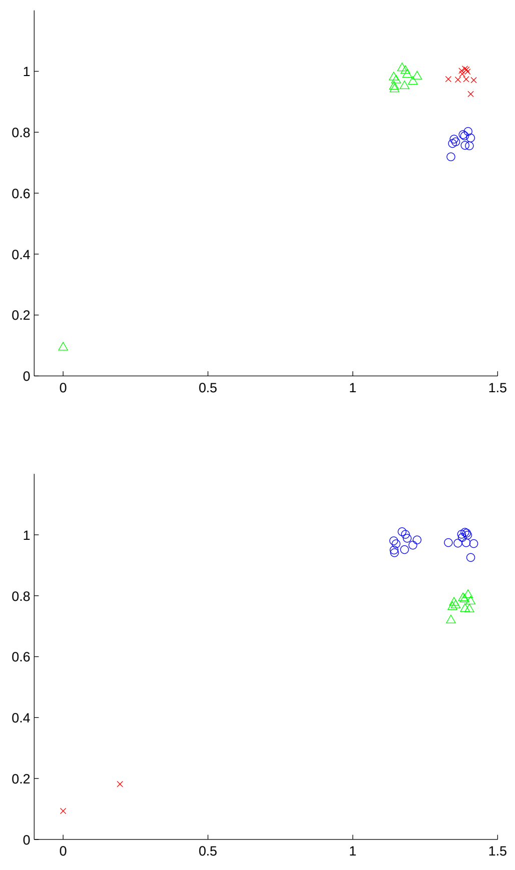
This formulation re-expresses the problem in a form that can be solved by conventional semidefinite programming techniques (Helmberg, 2000; Boyd and Vandenberghe, 2004).

4.2 Unsupervised Experimental Results

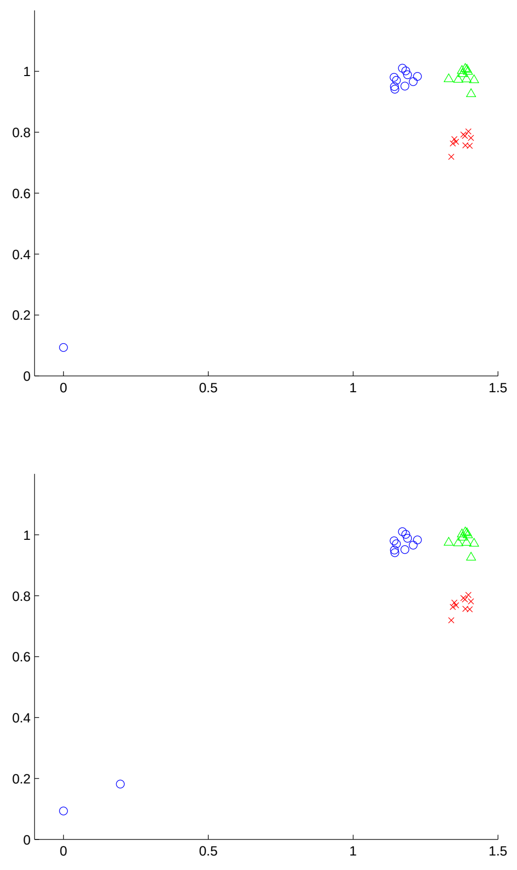
Based on the above semidefinite formulation for the multi-class unsupervised SVM training (4.8), I implemented the training procedures using the semidefninte programming package SDPT3 (Toh et al., 1999).[3]

First, I conducted some simple experiments to investigate the basic properties of spectral clustering methods in comparison to the approach developed above. In particular, I compared the proposed approach to the multi-class spectral clustering algorithm of (Ng et al., 2001). As described in Section 3.3, we expect that the normalized cut criterion used by (Ng et al., 2001) retains sensitivity to outliers. Figure 4.1 shows the results of an experiment on data embedded on the unit sphere, using the linear kernel as in Section 3.4. Here we see that the behavior of normalized cut is clearly influenced by the presence of outliers. However, there is another potential weakness: the technique of (Ng et al., 2001) uses k-means clustering to partition the data after it has been embedded in the space of the top κ eigenvectors, which means that its final clustering will tend to assume Gaussian (i.e. spherical) classes in feature space. Figure 4.2 demonstrates that that the maximum margin approach is indeed able to capture linear, but non-Gaussian class structure, whereas spectral clustering fails to preserve the intrinsic structure of the data.

[3]In general SDPT3 is faster than SeDuMi, and it is able to solve larger problems using the same memory.

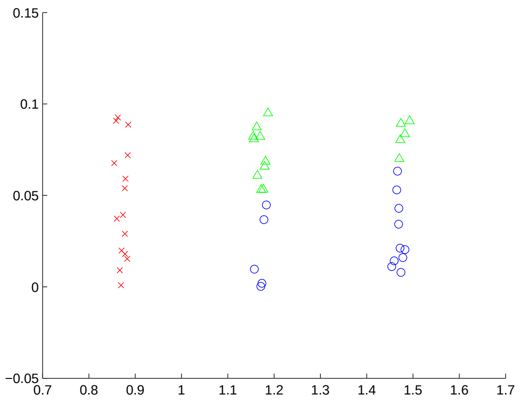


(a)

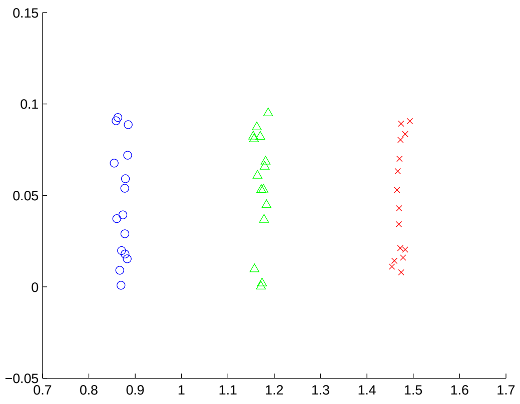


(b)

Figure 4.1: Normalized cut vs. maximum margin clustering in the multi-class case. (a) and (b) show the results produced by normalized cut and maximum margin clustering respectively. The two axes show angular positions on the unit sphere in spherical (ϕ, θ) coordinates.



(a)



(b)

Figure 4.2: Normalized cut vs. maximum margin clustering in the multi-class case. (a) and (b) show the results produced by normalized cut and maximum margin clustering respectively. The two axes show angular positions on the unit sphere in spherical (ϕ, θ) coordinates.

Note that, once again, even though the spectral clustering approach of (Ng et al., 2001) might avoid these specific problems on these data sets by using an appropriate kernel, it will always be sensitive to outliers and assume Gaussian classes in whatever feature space is implied by the kernel.

In the next set of experiments I compared the semidefinite multi-class clustering technique to k-means and the spectral clustering method of (Ng et al., 2001). Initial experiments were conducted on the sythetic data sets depicted in Figure 4.3. Table 4.1 shows that for the first four sets of data (AAAI, Circle&Balls, 3JoinedCircles, Squiggles) semidefinite and spectral clustering obtained identically small error rates, which were in turn significantly smaller than those obtained by k-means (except for the AAAI case).

The clustering algorithms were also evaluated on two real data sets consisting of images of hand-written digits; see Figures 4.4 and 4.5. The first data set, DigitsA, consists of black and white images of 20×16 pixels, as shown in Figure 4.4. The second data set, DigitsB, consists of grey scale images of 16×16 pixels, as shown in Figure 4.5. In Table 4.1, the rows with DigitsA are the results on the black and white digit data set, and the rows with DigitsB are the results on the grey scale digit data set. The digits included in the data sample are explicitly indicated. DigitsA* reports one run using all 39 examples of each digit while DigitsA reports 10 repeats of subsampling 20 out of 39 examples of each digit. DigitsB reports 10 repeats of subsampling 30 out of 1100 examples of each digit. Table 4.1 shows that semidefinite clustering demonstrates a significant advantage over k-means clustering on these data sets, but also a systematic advantage over spectral clustering.

The results shown above demonstrate that the large margin principle of the unsupervised SVM approach is very effective discovering complex clusterings in the data. Moreover, the advantage of obtaining a universal classification rule remains in the multi-class case, which enables us to classify future data without re-running the algorithm.

4.3 Semi-supervised Multi-class SVMs

Again, the above formulation of *unsupervised* multi-class SVM training can easily be extended to *semi-supervised* learning. The extension of the unsupervised training procedure

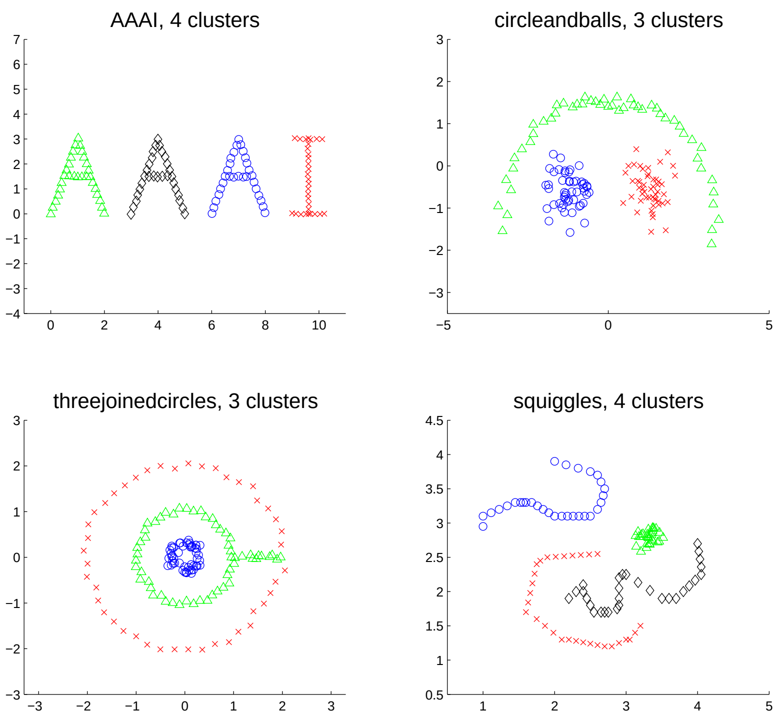


Figure 4.3: Four artificial data sets used in the clustering experiments.

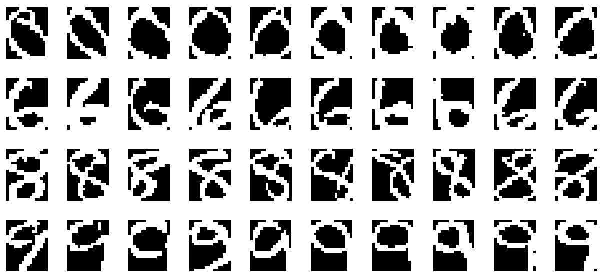


Figure 4.4: A sampling of handwritten digit images from the first data set, DigitsA (zeros, sixes, eights and nines).

Clustering:	**Semidefinite**	Spectral	Kmeans
AAAI	**0**	0	0
Circle&Balls	**0**	0	18.7
3Circles	**4.0**	4.0	47.3
Squiggles	**0**	0	28.3
DigitsA689*	**3.4**	12.0	12.0
DigitsA689	**7.2 ±1.3**	12.3 ±2.3	14.0 ±4.4
DigitsB689	**13.3 ±2.8**	15.1 ±3.9	18.5 ±4.6
DigitsA0689*	**7.5**	9.2	38.3
DigitsA0689	**11.6 ±1.8**	20.4 ±2.4	24.1 ±4.1
DigitsB0689	**21.1 ±2.3**	25.8 ±2.6	27.7 ±4.4

Table 4.1: Clustering results: Percentage misclassification errors of the various clustering algorithms on the various data sets (± one standard error of the mean).

to the semi-supervised case proceeds similarly to the two-class case. Here one simply adds the constraints given by the observed labels to the semidefinite program: $M_{ij} = \delta_{y^i,y^j}$ for all labeled training pairs i, j, and $D_{ir} = \delta_{y^i,r}$ for each labeled training pair and class label i, r. The remainder of the program is equivalent to (4.8), and thus still defines a semidefinite programming problem and can be solved by standard methods.

4.4 Comparison to Existing Semi-supervised Methods

Unlike the unsupervised case where multi-class k-means and spectral clustering are still applicable, in the semi-supervised scenario, to the best of my knowledge, there are no other procedures available for multi-class SVMs that apply the large margin approach of (Joachims, 1999; Bennett and Demiriz, 1998), therefore, standard multi-class SVMs (Crammer and Singer, 2001) are compared to, where the labeled training part is taken as the whole training data.

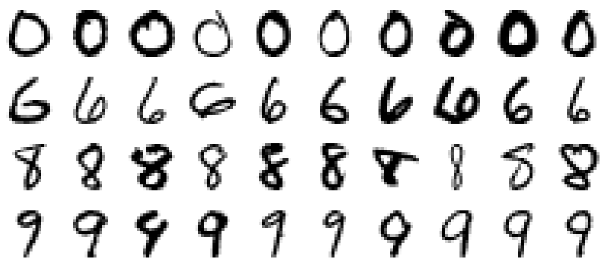


Figure 4.5: A sampling of handwritten digit images from the second data set, DigitsB (zeros, sixes, eights and nines).

4.5 Semi-supervised Experimental Results

In the semi-supervised case, I tested my approach to semi-supervised learning on various data sets. As mentioned, there are no other semi-supervised training techniques based on the large margin principle or the semidefinite approach we develop above. Therefore, I compare to standard *supervised* multi-class SVMs (Crammer and Singer, 2001) but also to a general semi-supervised approach based on clustering the data using spectral clustering, and then using the labeled data to assign minimum error labels to the classes. In each case, I evaluated the techniques transductively as discribed in Section 3.7.

In these experiments, I used two UCI data sets (Balance-scale and Cars), one of the hand-written digits data sets (DigitsA), and a data set of face images. The results were averaged over 10 repeats, each randomly subsampling 1/10 of the data to be labeled (Table 4.2). Here we see that the semi-supervised semidefinite approach tends to outperform the other methods, particularly on the digits data. Generally, semi-supervised training shows an advantage over strictly supervised training which ignores the extra unlabeled data. Table 4.2 shows that the semidefinite SVM method is effective at exploiting unlabeled data to improve the prediction of held out labels. In every case (except 5faces), it significantly reduces the error of standard multi-class SVM, and obtains the best overall performance of the semi-supervised learning techniques we have investigated. (See the Table 4.2 caption for details of the experimental set up.)

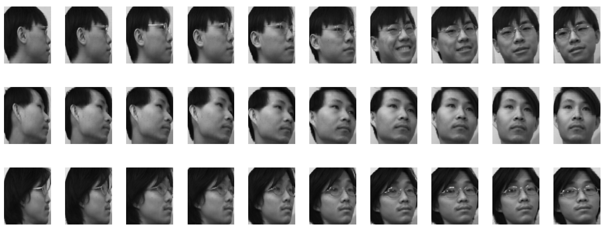


Figure 4.6: A sampling of the face data (three people).

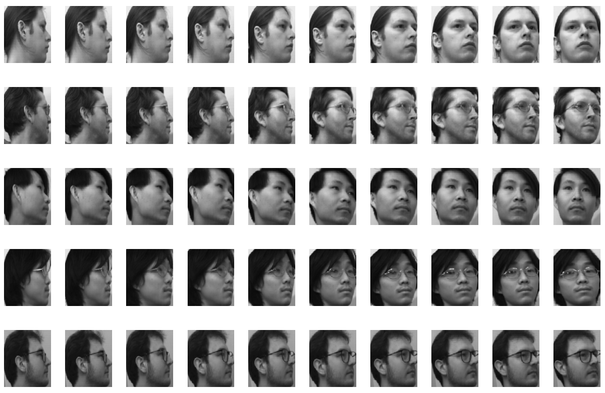


Figure 4.7: A sampling of the face data (five people).

Semi-sup:	**Semidefinite**	supSVM	Spectral
UCI: Balance-scale	**24.1 ±1.7**	30.3 ±2.4	26.3 ±4.3
UCI:Cars	**41.9 ±0.4**	43.1 ±1.4	43.2 ±0.4
DigitsA689	**5.6 ±0.7**	16.3 ±1.6	12.7 ±0.3
DigitsA0689	**6.5 ±0.5**	20.6 ±3.3	11.8 ±2.7
3faces	**0**	10.0 ±2.0	0
5faces	**5.2 ±1.5**	14.4 ±1.7	0

Table 4.2: Semi-supervised learning results: Percentage misclassification errors of the various semi-supervised learning algorithms on the various data sets (± one standard error of the mean). Data set sizes are: UCI:Balance-scale: 3 classes, 40 examples each. UCI:Cars: 3 classes, 40 examples each. DigitsA689: 3 classes, 39 examples each. DigitsA0689: 4 classes, 39 examples each. 3faces: 3 classes, 32 examples each. 5faces: 5 classes, 30 examples each.

4.6 Conclusion

In this chapter, the learning techniques of unsupervised and semi-supervised two-class SVMs, presented in the previous chapter, have been generalized and extended to the multi-class case. The methodologies share the same framework, which is to take the dual of the maximum margin problem, re-express it in terms of the indicator variables that compare the y-labels and then relax the indicator variables to get a convex optimization problem. However, the derivation here involves more technical complications. Experiments show that the proposed methods produce results that are competitive with, and sometimes better than the state of the art. Overall, semidefinite programming has shown itself to be an effective approach for unifying and generalizing SVM training techniques, and applying SVMs to a wider range of circumstances, like unsupervised and semi-supervised learning. Moreover, as shown in the background, the supervised multivariate prediction problem can be viewed as a special case of multi-class classification problem, therefore, with the multi-class unsupervised learning technique, the further extension to unsupervised *multivariate* problems becomes possible.

Chapter 5

Unsupervised Structured-output Support Vector Machines

There have recently been a number of significant advances in learning structured multivariate predictors from labeled data (Tsochantaridis et al., 2004; Altun et al., 2003; Taskar et al., 2003). Structured prediction extends the standard supervised learning framework beyond the univariate setting, where a single output variable is considered, to the multivariate setting, where complex, non-scalar predictions $\hat{\mathbf{y}} = f(\mathbf{x})$ must be produced for inputs $\mathbf{x}$. The challenge this creates is that each component $\hat{y}_i$ of $\hat{\mathbf{y}}$ should not be produced in isolation, depending only on the input $\mathbf{x}$, but instead should take into account correlations or constraints between $\hat{y}_i$ and its neighboring components $\hat{y}_j \in \hat{\mathbf{y}}$. It has been shown in many applications that structured predictors that directly capture the relationships between output components perform better than models that do not directly enforce these relationships (Tsochantaridis et al., 2004; Altun et al., 2003; Taskar et al., 2003). However, recent progress on learning structured predictors has focused primarily on the *supervised* case, where the structured output labels are provided with the training data.

In this chapter, I extend an important existing technique for learning structured multivariate predictors to the *unsupervised* case, by generalizing the approach I have developed in Chapter 3 and 4. Although the technique I develop is general, in this chapter I will focus the exposition on the special case of hidden Markov models (HMMs) that was introduced in Section 2.3.

As introduced in Section 2.3, an HMM expresses a joint distribution over an observation-sequence state-sequence pair $(\mathbf{x}, \mathbf{y})$. A significant appeal of these models is that nearly every operation one might wish to perform with them is tractable. For example, as discussed in Section 2.3.1, an HMM can be used to efficiently decode an observation sequence to recover an optimal hidden state sequence, $\mathbf{y}^* = \arg\max_{\mathbf{y}} p(\mathbf{y}|\mathbf{x})$, by the Viterbi algorithm (Rabiner, 1989). Another example is maximum likelihood training discussed in Section 2.3.2: if one is given *complete* training data expressed in paired sequences $(\mathbf{x}^1, \mathbf{y}^1), ..., (\mathbf{x}^n, \mathbf{y}^n)$ then training a hidden Markov model to maximize joint likelihood is a trivial matter of setting the observation and transition potentials to the observed frequency counts of each state$\rightarrow$state, state$\rightarrow$observation, and initial state patterns.

More often, however, as noted in Section 2.3.2, people are more interested in learning a conditional model $p(\mathbf{y}|\mathbf{x})$ rather than a joint model $p(\mathbf{x}, \mathbf{y})$, because the conditional model is needed for structured prediction in subsequent applications. For structured prediction applications, the performance of discriminative training techniques, that optimize $p(\mathbf{y}|\mathbf{x})$ generally exceeds that of more straightforward joint training techniques, that optimize $p(\mathbf{x}, \mathbf{y})$.

One major limitation of current discriminative training algorithms, however, is that they are all *supervised.* That is, these techniques require complete state sequences $\mathbf{y}$ to be provided with the observation sequences $\mathbf{x}$, which precludes the use of unsupervised or semi-supervised approaches. This is a serious limitation because in most application areas of sequence processing—be it speech, language, or biological sequence analysis—labeled state sequence information is very hard or expensive to obtain as in the univariate case, whereas unlabeled observation sequences are very cheap and available in almost unlimited supply. Intuitively, much of the state-class structure of the domain can already be inferred from a massive collection of unlabeled observation sequences. My goal in this chapter is to try to develop a generally effective technique for unsupervised learning of discriminative models. Such techniques do not currently exist.[1]

As discussed in Section 2.3.3, for the *unsupervised* training of hidden Markov models, most researchers back off to a joint model view, and use EM to recover a conditional

[1]An exception is (Altun et al., 2005), which considers a *semi-supervised* training approach. Unfortunately, the technique is not applicable to the unsupervised case I address in this chapter.

model as a side-effect of acquiring $p(\mathbf{x}, \mathbf{y})$. It is interesting to note that, even given the advantage of discriminative versus joint training for the supervised case, most research on unsupervised training of HMMs has had to drop the discriminative approach. However, there are several well-known problems with EM that I point out in Section 2.3.3. First, EM is *not* an efficient optimization technique. That is, the marginal likelihood criterion it attempts to optimize is not concave, and EM only converges to local maxima. Thus, from the global optimization perspective, EM fails to guarantee a solution to the problem. Second, if we are interested in learning a discriminative model $p(\mathbf{y}|\mathbf{x})$, there is little reason to expect that the training criterion used by EM, which focuses on improving the input model $p(\mathbf{x})$, will recover a good predictor. In fact, given the experience with discriminative versus joint supervised training, there is every reason to expect that it will not.

This leads to the work I pursue in this chapter: we will see that it is possible to optimize a *discriminative* training criterion even when learning an HMM model in the *unsupervised* case. I will also show that it is possible to achieve this in a convex optimization framework, where at least in principle it is possible to compute optimal solutions in polynomial time (Nesterov and Nemirovskii, 1994). In particular, I will extend the supervised maximum margin Markov network ($\mathrm{M}^3\mathrm{N}$) approach outlined in Section 2.3.2, and show that even without training labels (hidden state sequences) one can still learn a discriminative model $p(\mathbf{y}|\mathbf{x})$ that postulates "widely separated" hidden state sequences for different input observation sequences.

5.1 Discriminative Unsupervised Training

To develop a discriminative unsupervised training approach for structured predictors, I follow the same methodology outlined in Chapter 3 and 4 which involves the same sequence of steps: First, one treats the missing training labels as variables to be optimized. Second, one takes the dual of the supervised problem, yielding an expression that involves pairwise comparisons between the postulated y-labels. Third, one re-expresses the problem in terms of comparisons between y-labels, rather than the y-labels themselves, which yields a convex function in the comparison variables, even when the objective was not convex in the original y-labels. Then, one relaxes the comparison variables to real values, and possibly relaxes additional constraints, to obtain a convex optimization problem. Finally, the comparison variables in the solution can be used to recover a classification of the original data.

5.2 Unsupervised M^{3}Ns

As usual, the development of an unsupervised training technique begins with the supervised case, which in this case is the M^{3}N approach discussed in Section 2.3.2. Although the presentation here will be focused on extending M^{3}N for sequence predictions, the ideas easily extend to other approaches, and other structured prediction problems.

Recall the representation of the supervised problem established in Section 2.3.2. Suppose we are given labeled training sequences $(\mathbf{x}^1, \mathbf{y}^1), ..., (\mathbf{x}^n, \mathbf{y}^n)$, where each individual training sequence is of the form $(\mathbf{x}^i = (x^i_1, ..., x^i_L), \mathbf{y}^i = (y^i_1, ..., y^i_L))$. As before, we consider feature functions $\boldsymbol{\phi}(\mathbf{x}^i, \mathbf{y}^i)$ of both the observation and the label sequence. If one assumes a stationary hidden Markov model, as we do, then (2.44) and (2.45) show that the vector of features $\boldsymbol{\phi}(\mathbf{x}^i, \mathbf{y}^i)$ can be re-expressed as a sum of feature vectors over the local pieces of the example, which I rewrite here for convenience

$$\boldsymbol{\phi}(\mathbf{x}^i, \mathbf{y}^i) \quad = \quad \sum_{k=1}^{L} \boldsymbol{\phi}(x^i_k y^i_k y^i_{k-1})$$

That is, each feature vector $\boldsymbol{\phi}(x^i_k y^i_k y^i_{k-1})$ for a local sequence piece $(x^i_k y^i_k y^i_{k-1})$ is just a sparse vector that indicates which particular configuration is true in each local table of the graphical HMM model. The fact that the feature vector depends only on a local subset of variables $(x^i_k y^i_k y^i_{k-1})$ encodes the conditional independence assumption of the HMM. The fact that the total feature vector for a complete labeled sequence is just a sum of the feature vectors for each local sequence piece, independent of k, encodes the stationarity assumption of the HMM.

As discussed in Section 2.3.2, this would be the representation one would use in training a supervised M^{3}N given labeled training sequences. In the supervised case, a discriminative structured predictor can be trained by solving a quadratic program (2.47) with respect to weights on the feature vector $\boldsymbol{\phi}(\mathbf{x}, \mathbf{y})$. The goal of this quadratic program is to minimize the multivariate margin loss

$$\omega(\mathbf{y}^1, ..., \mathbf{y}^n) = \min_{\mathbf{w}} \frac{\beta}{2}\|\mathbf{w}\|^2 + \sum_{i,\mathbf{u}} \max_{\mathbf{u}} \Big[\Delta(\mathbf{u}, \mathbf{y}^i) - \mathbf{w} \cdot (\boldsymbol{\phi}(\mathbf{x}^i, \mathbf{y}^i) - \boldsymbol{\phi}(\mathbf{x}^i, \mathbf{u})) \Big]_+ \tag{5.1}$$

where $\Delta(\mathbf{u}, \mathbf{y}^i)$ counts the disagreements between $\mathbf{u}$ and $\mathbf{y}^i$, thus encouraging a margin between the correct labeling $\mathbf{y}^i$ and an alternative labeling $\mathbf{u}$ that is proportional to their Hamming distance.

To derive a convenient unsupervised form of the training criterion, we will need to work with the dual quadratic program (2.48). Taskar et al. (2003) observed that, as discussed in Section 2.3.2 the dual QP, although of exponential size in a naive derivation, can be factored into marginal dual variables using the conditional independence structure of the y-labeling which can yield a compact formulation of the dual as a polynomial sized quadratic program. In the HMM representation we are assuming, a compact quadratic program that computes the same multivariate margin loss can be expressed as (2.49), which I repeat here for convenience

$$\begin{aligned}\omega(\mathbf{y}^1, ..., \mathbf{y}^n) \quad = \quad & \max_{\boldsymbol{\mu},\boldsymbol{\nu}} \sum_{i,k,u} \mu_{ik}(u) 1_{(u \neq y^i_k)} \\ & \qquad -\frac{1}{2\beta} \sum_{ij,k\ell,uu',vv'} \nu_{ik}(uu')\nu_{j\ell}(vv')\Delta\boldsymbol{\phi}_{ik}(uu')^\top \Delta\boldsymbol{\phi}_{j\ell}(vv') \qquad (5.2) \\ \text{subject to} \quad & \mu_{ik}(u) \geq 0, \forall iku \\ & \nu_{ik}(uu') \geq 0, \forall ikuu' \\ & \sum_{u'} \nu_{ik}(uu') = \mu_{ik}(u), \quad \sum_u \mu_{ik}(u) = 1\end{aligned}$$

where $\Delta\boldsymbol{\phi}_{ik}(uu') = \boldsymbol{\phi}(x^i_k y^i_k y^i_{k-1}) - \boldsymbol{\phi}(x^i_k uu')$. Here i, j index training cases, k, ℓ index locations in each training sequence, and u, u' index possible relabelings at these locations. There is a dual variable $\mu_{ik}(u)$ corresponding to each singleton relabeling, and a dual variable $\nu_{ik}(uu')$ corresponding to each adjacent pair relabeling.

To derive an unsupervised version of this training criterion, we now consider minimizing the multivariate margin loss ω as a function of the sequence labelings $\mathbf{y}^1, ..., \mathbf{y}^n$. Our task will be made considerably simpler by reformulating the multivariate margin loss in terms of the indicator and equivalence relation matrices that we will ultimately have to use.

Theorem 4 *The quadratic terms in (5.2) can be reformulated as*

$$\sum_{ij,k\ell,uv} \mu_{ik}(u)\mu_{j\ell}(v)\,\delta_1(iku,j\ell v)K(ik,j\ell) + \sum_{ij,k\ell,uu',vv'} \nu_{ik}(uu')\nu_{j\ell}(vv')\delta_2(ikuu',j\ell vv') \quad (5.3)$$

$$\begin{aligned} \textit{where}\quad \delta_1(iku,j\ell v) &= 1_{(y_k^i=y_\ell^j)} - 1_{(u=y_\ell^j)} - 1_{(y_k^i=v)} + 1_{(u=v)} \\ \delta_2(ikuu',j\ell vv') &= 1_{(y_{k,k-1}^i=y_{\ell,\ell-1}^j)} - 1_{(uu'=y_{\ell,\ell-1}^j)} - 1_{(y_{k,k-1}^i=vv')} + 1_{(uu'=vv')} \end{aligned}$$

Here $K(ik,j\ell)$ is the inner product between the sub-feature vectors that omit the transition model features and set the current state values equal.

Proof Since we have $\Delta\boldsymbol{\phi}_{ik}(uu') = \boldsymbol{\phi}_{ik}(y_{k,k-1}^i) - \boldsymbol{\phi}_{ik}(uu'), \Delta\boldsymbol{\phi}_{j\ell}(vv') = \boldsymbol{\phi}_{jl}(y_{\ell,\ell-1}^j) - \boldsymbol{\phi}_{jl}(vv')$, the quadratic term $\Delta\boldsymbol{\phi}_{ik}(uu')^\top\Delta\boldsymbol{\phi}_{j\ell}(vv')$ can be reformulated into

$$\begin{aligned} & \Delta\boldsymbol{\phi}_{ik}(uu')^\top\Delta\boldsymbol{\phi}_{j\ell}(vv') \\ = \; & \boldsymbol{\phi}_{ik}(y_{k,k-1}^i)^\top\boldsymbol{\phi}_{j\ell}(y_{\ell,\ell-1}^j) \\ - \; & \boldsymbol{\phi}_{ik}(uu'))^\top\boldsymbol{\phi}_{j\ell}(y_{\ell,\ell-1}^j) \\ - \; & \boldsymbol{\phi}_{ik}(y_{k,k-1}^i)^\top\boldsymbol{\phi}_{j\ell}(vv')) \\ + \; & \boldsymbol{\phi}_{ik}(uu'))^\top\boldsymbol{\phi}_{j\ell}(vv')) \end{aligned}$$

Here to make things simple, we only consider linear feature vectors. From the arguments shown in (2.42), we can see that for each local sequence piece $(x_k^i y_k^i y_{k-1}^i)$, the feature vector $\phi_{ik}(y_{k,k-1}^i)$ is just a sparse vector that indicates which particular configuration is true in each local table of the graphical HMM model. More specifically, the feature vector can be written as $[\mathbf{1}_{x_k^i y_k^i};\mathbf{1}_{y_k^i y_{k-1}^i}]$, where $\mathbf{1}_{x_k^i y_k^i}$ is a vector that enumerates all configurations of $(x^i y^i)$, all the entries except the one for $(x_k^i y_k^i)$ are zero.

Therefore, we have

$$\begin{aligned} \boldsymbol{\phi}_{ik}(y_{k,k-1}^i)^\top\boldsymbol{\phi}_{j\ell}(y_{\ell,\ell-1}^j) &= 1_{(y_k^i x_k^i = y_\ell^j x_\ell^j)} + 1_{(y_{k,k-1}^i=y_{\ell,\ell-1}^j)} \\ &= 1_{(y_k^i=y_\ell^j)}K(ik,j\ell) + 1_{(y_{k,k-1}^i=y_{\ell,\ell-1}^j)} \end{aligned} \quad (5.4)$$

If we rewrite the other three terms similarly, the quadratic terms in (5.2) can be reformulated into

$$\begin{aligned}
& \textstyle\sum_{ij,k\ell,uu',vv'} \nu_{ik}(uu')\nu_{j\ell}(vv')\Big[\Big(1_{(y^i_k=y^j_\ell)}K(ik,j\ell) + 1_{(y^i_{k,k-1}=y^j_{\ell,\ell-1})}\Big) \\
& \qquad - \Big(1_{(u=y^j_\ell)}K(ik,j\ell) + 1_{(uu'=y^j_{\ell,\ell-1})}\Big) \\
& \qquad - \Big(1_{(y^i_k=v)}K(ik,j\ell) + 1_{(y^i_{k,k-1}=vv')}\Big) \\
& \qquad - \big(1_{(u=v)}K(ik,j\ell) + 1_{(uu'=vv')}\big) \Big] \\
= \;& \textstyle\sum_{ij,k\ell,uu',vv'} \nu_{ik}(uu')\nu_{j\ell}(vv') \Big(1_{(y^i_k=y^j_\ell)} - 1_{(u=y^j_\ell)} - 1_{(y^i_k=v)} + 1_{(u=v)}\Big) K(ik,j\ell) \\
& + \textstyle\sum_{ij,k\ell,uu',vv'} \nu_{ik}(uu')\nu_{j\ell}(vv') \Big(1_{(y^i_{k,k-1}=y^j_{\ell,\ell-1})} - 1_{(uu'=y^j_{\ell,\ell-1})} - 1_{(y^i_{k,k-1}=vv')} + 1_{(uu'=vv')}\Big) \\
= \;& \textstyle\sum_{ij,k\ell,uu',vv'} \nu_{ik}(uu')\nu_{j\ell}(vv')\delta_1(iku,j\ell v)K(ik,j\ell) + \sum_{ij,k\ell,uu',vv'} \nu_{ik}(uu')\nu_{j\ell}(vv')\delta_2(ikuu',j\ell vv') \\
= \;& \textstyle\sum_{ij,k\ell,u,v} \mu_{ik}(u)\mu_{j\ell}(v)\delta_1(iku,j\ell v)K(ik,j\ell) + \sum_{ij,k\ell,uu',vv'} \nu_{ik}(uu')\nu_{j\ell}(vv')\delta_2(ikuu',j\ell vv')
\end{aligned}$$

because $\sum_{u'} \nu_{ik}(uu') = \mu_{ik}(u)$ ∎

Then, we can define the indicator matrices M, N, C, D

$$\begin{aligned}
M_{ik,j\ell} &= 1_{(y^i_k=y^j_\ell)} \\
N_{ikk-1,j\ell\ell-1} &= 1_{(y^i_{k,k-1}=y^j_{\ell,\ell-1})} \\
C_{ik,u} &= 1_{(y^i_k=u)} \\
D_{ikk-1,uu'} &= 1_{(y^i_{k,k-1}=uu')}
\end{aligned} \qquad (5.5)$$

Note that by these definitions, M and N are equivalence relations on singleton and pairwise positions in the y-label sequences, respectively. Also, by these definitions $M = CC^\top$ and $N = DD^\top$. We can now also place the optimization variables into corresponding matrices $\boldsymbol{\mu}$ and $\boldsymbol{\nu}$ such that $\boldsymbol{\mu}_{ik,u} = \mu_{ik}(u)$ and $\boldsymbol{\nu}_{ik,uu'} = \nu_{ik}(uu')$. Given these definitions, we can then re-express an equivalent quadratic program to (5.2) in a compact matrix form. Letting p_1 be the number of singleton positions in the training data, and letting E denote the matrix of all 1's, we obtain

Theorem 5 *The multivariate margin loss (5.2) equals*

$$\begin{aligned}
\omega(M,N,C,D) \quad &= \quad p_1 - \langle \boldsymbol{\mu}, C\rangle + \tfrac{1}{\beta}(\langle \boldsymbol{\mu}, KC\rangle + \langle \boldsymbol{\nu}, ED\rangle) \\
&\qquad - \tfrac{1}{2\beta}\left(\langle M, K\rangle + \langle N, E\rangle + \langle \boldsymbol{\mu}\boldsymbol{\mu}^\top, K\rangle + \langle \boldsymbol{\nu}\boldsymbol{\nu}^\top, E\rangle\right) \\
\textit{subject to} \quad & \textstyle\sum_{u'} \nu_{ik}(uu') = \mu_{ik}(u) \quad \forall iku \\
& \boldsymbol{\mu} \geq 0,\ \boldsymbol{\nu} \geq 0,\ \boldsymbol{\nu}\mathbf{e} = \mathbf{e} \\
& M = CC^\top,\ N = DD^\top \\
& N_{ikk-1,j\ell\ell-1} = M_{ik,j\ell}M_{ik-1,j\ell-1} \quad \forall ijk\ell
\end{aligned} \tag{5.6}$$

Proof First we rewrite the linear term of the multivariate margin loss (5):

$$\begin{aligned}
&\textstyle\sum_{i,k,u} \mu_{ik}(u) 1_{(u \neq y_k^i)} \\
&= \textstyle\sum_{i,k,u} \mu_{ik}(u)(1 - 1_{(u = y_k^i)}) \\
&= \textstyle\sum_{i,k,u} \mu_{ik}(u) - \sum_{i,k,u} \boldsymbol{\mu}_{ik,u} C_{ik,u} \\
&= \textstyle\sum_{i,k} 1 - \langle \boldsymbol{\mu}, C\rangle \\
&= p_1 - \langle \boldsymbol{\mu}, C\rangle
\end{aligned} \tag{5.7}$$

Second, it has been proven for Theorem 4 that the quadratic term in (5) can be reformulated as (5.3), which can be further decomposed into

$$\sum_{ij,k\ell,u,v} \mu_{ik}(u)\mu_{j\ell}(v) 1_{(y_k^i = y_\ell^j)} K(ik, j\ell) \tag{5.8}$$

$$- \sum_{ij,k\ell,u,v} \mu_{ik}(u)\mu_{j\ell}(v) 1_{(u = y_\ell^j)} K(ik, j\ell) \tag{5.9}$$

$$- \sum_{ij,k\ell,u,v} \mu_{ik}(u)\mu_{j\ell}(v) 1_{(y_k^i = v)} K(ik, j\ell) \tag{5.10}$$

$$+ \sum_{ij,k\ell,u,v} \mu_{ik}(u)\mu_{j\ell}(v) 1_{(u = v)} K(ik, j\ell) \tag{5.11}$$

$$+ \sum_{ij,k\ell,uu',vv'} \nu_{ik}(uu')\nu_{j\ell}(vv') 1_{(y_{k,k-1}^i = y_{\ell,\ell-1}^j)} \tag{5.12}$$

$$- \sum_{ij,k\ell,uu',vv'} \nu_{ik}(uu')\nu_{j\ell}(vv')1_{(uu'=y^j_{\ell,\ell-1})} \tag{5.13}$$

$$- \sum_{ij,k\ell,uu',vv'} \nu_{ik}(uu')\nu_{j\ell}(vv')1_{(y^i_{k,k-1}=vv')} \tag{5.14}$$

$$+ \sum_{ij,k\ell,uu',vv'} \nu_{ik}(uu')\nu_{j\ell}(vv')1_{(uu'=vv')} \tag{5.15}$$

Below I will rewrite each component from (5.8)-(5.15) in terms of the indicator matrices M, N, C and D.
(5.8):

$$\begin{aligned}
&\sum_{ij,k\ell,u,v} \mu_{ik}(u)\mu_{j\ell}(v)1_{(y^i_k=y^j_\ell)}K(ik,j\ell) \\
&= \sum_{ij,k\ell} \left(\sum_u \mu_{ik}(u)\right)\left(\sum_v \mu_{j\ell}(v)\right) M_{ik,j\ell}K_{ik,j\ell} \\
&= \sum_{ij,k\ell} M_{ik,j\ell}K_{ik,j\ell} \\
&= \langle M, K\rangle
\end{aligned} \tag{5.16}$$

(5.9):

$$\begin{aligned}
&\sum_{ij,k\ell,u,v} \mu_{ik}(u)\mu_{j\ell}(v)1_{(u=y^j_\ell)}K(ik,j\ell) \\
&= \sum_{ij,k\ell,u} \mu_{ik}(u)\left(\sum_v \mu_{j\ell}(v)\right) 1_{(u=y^j_\ell)}K(ik,j\ell) \\
&= \sum_{ij,k\ell,u} \boldsymbol{\mu}_{ik,u}C_{j\ell,u}K_{ik,j\ell} \\
&= \sum_{i,k,u} \boldsymbol{\mu}_{ik,u}\,(KC)_{ik,u} \\
&= \langle \boldsymbol{\mu}, KC\rangle
\end{aligned} \tag{5.17}$$

(5.10), similar to (5.9)

$$\sum_{ij,k\ell,u,v} \mu_{ik}(u)\mu_{j\ell}(v)1_{(y^i_k=v)}K(ik,j\ell)$$

$$\begin{aligned}
&= \sum_{ij,k\ell,v} \mu_{j\ell}(v) 1_{(y^i_k=v)} K(ik,j\ell) \\
&= \sum_{ij,k\ell,v} \boldsymbol{\mu}_{j\ell,v} C_{ik,v} K_{ik,j\ell} \\
&= \langle \boldsymbol{\mu}, KC \rangle \qquad (5.18)
\end{aligned}$$

(5.11):

$$\begin{aligned}
&\sum_{ij,k\ell,u,v} \mu_{ik}(u)\mu_{j\ell}(v) 1_{(u=v)} K(ik,j\ell) \\
&= \sum_{ij,k\ell,u} \mu_{ik}(u)\mu_{j\ell}(u) K(ik,j\ell) \\
&= \sum_{ij,k\ell,u} \boldsymbol{\mu}_{ik,u}\boldsymbol{\mu}_{j\ell,u} K(ik,j\ell) \\
&= \sum_{ij,k\ell} \left(\boldsymbol{\mu}\boldsymbol{\mu}^\top\right)_{ik,j\ell} K(ik,j\ell) \\
&= \langle \boldsymbol{\mu}\boldsymbol{\mu}^\top, K \rangle \qquad (5.19)
\end{aligned}$$

(5.12):

$$\begin{aligned}
&\sum_{ij,k\ell,uu',vv'} \nu_{ik}(uu')\nu_{j\ell}(vv') 1_{(y^i_{k,k-1}=y^j_{\ell,\ell-1})} \\
&= \sum_{ij,k\ell} \left(\sum_{uu'} \nu_{ik}(uu')\right)\left(\sum_{vv'} \nu_{j\ell}(vv')\right) 1_{(y^i_{k,k-1}=y^j_{\ell,\ell-1})} \\
&= \sum_{ij,k\ell} N_{ikk-1,j\ell\ell-1} \\
&= \langle N, E \rangle \qquad (5.20)
\end{aligned}$$

(5.13):

$$\begin{aligned}
&\sum_{ij,k\ell,uu',vv'} \nu_{ik}(uu')\nu_{j\ell}(vv') 1_{(uu'=y^i_{\ell,\ell-1})} \\
&= \sum_{ij,k\ell,uu'} \nu_{ik}(uu') \left(\sum_{vv'} \nu_{j\ell}(vv')\right) 1_{(uu'=y^j_{\ell,\ell-1})}
\end{aligned}$$

$$\begin{aligned}
&= \sum_{ij,k\ell,uu'} \boldsymbol{\nu}_{ik,uu'} D_{j\ell\ell-1,uu'} \\
&= \sum_{ij,k\ell,uu'} \boldsymbol{\nu}_{ik,uu'} E_{ik,j\ell\ell-1} D_{j\ell\ell-1,uu'} \\
&= \sum_{ik,uu'} \boldsymbol{\nu}_{ik,uu'} \left(ED\right)_{ik,uu'} \\
&= \langle \boldsymbol{\nu}, ED \rangle \qquad (5.21)
\end{aligned}$$

(5.14), similar to (5.13)

$$\begin{aligned}
&\sum_{ij,k\ell,uu',vv'} \nu_{ik}(uu')\nu_{j\ell}(vv')1_{(y^i_{k,k-1}=vv')} \\
&= \sum_{ij,k\ell,vv'} \nu_{j\ell}(vv')1_{(y^i_{k,k-1}=vv')} \\
&= \sum_{ij,k\ell,vv'} \boldsymbol{\nu}_{j\ell,vv'} D_{ikk-1,vv'} \\
&= \langle \boldsymbol{\nu}, ED \rangle \qquad (5.22)
\end{aligned}$$

(5.15):

$$\begin{aligned}
&\sum_{ij,k\ell,uu',vv'} \nu_{ik}(uu')\nu_{j\ell}(vv')1_{(uu'=vv')} \\
&= \sum_{ij,k\ell,uu'} \nu_{ik}(uu')\nu_{j\ell}(uu') \\
&= \sum_{ij,k\ell,uu'} \boldsymbol{\nu}_{ik,uu'} \boldsymbol{\nu}_{j\ell,uu'} \\
&= \sum_{ij,k\ell} \left(\boldsymbol{\nu}\boldsymbol{\nu}^\top\right)_{ik,j\ell} \\
&= \langle \boldsymbol{\nu}\boldsymbol{\nu}^\top, E \rangle \qquad (5.23)
\end{aligned}$$

Combining (5.2) and (5.16)-(5.23) together and substituting into (5.2), we can rewrite the multivariate margin loss as

$$\begin{aligned}
\omega(M,N,C,D) \quad = \quad & p_1 - \langle \boldsymbol{\mu}, C \rangle + \tfrac{1}{\beta}(\langle \boldsymbol{\mu}, KC \rangle + \langle \boldsymbol{\nu}, ED \rangle) \\
& - \tfrac{1}{2\beta}\left(\langle M, K \rangle + \langle N, E \rangle + \langle \boldsymbol{\mu}\boldsymbol{\mu}^\top, K \rangle + \langle \boldsymbol{\nu}\boldsymbol{\nu}^\top, E \rangle\right)
\end{aligned} \qquad (5.24)$$

∎

With this matrix form of the multivariate margin loss we can now develop a convex optimization objective for discriminative unsupervised training of a structured prediction model. In particular, given unlabeled sequences $\mathbf{x}^1, ..., \mathbf{x}^n$, we would like to solve for the sequence labelings—represented implicitly as indicator matrices M, N, C and D over the singleton and pair labelings—by minimizing the multivariate margin loss of the resulting M^{3}N predictor. In formulating the optimization problem, there are a number of constraints on the matrix variables that we need to impose.

First, one needs to impose class balance constraints on the indicator matrices to avoid trivial results. This can be achieved by imposing class balance constraints on each local singleton labeling y_k^i and each local pairwise labeling $y_{k,k-1}^i$ by

$$\begin{array}{rcl} \left(\frac{1}{\kappa} - \epsilon\right) p_1\mathbf{e} \leq & M\mathbf{e} & \leq \left(\frac{1}{\kappa} + \epsilon\right) p_1\mathbf{e} \\ \left(\frac{1}{\kappa^2} - \epsilon\right) p_2\mathbf{e} \leq & N\mathbf{e} & \leq \left(\frac{1}{\kappa^2} + \epsilon\right) p_2\mathbf{e} \end{array} \tag{5.25}$$

where p_1 and p_2 are the number of singleton and pair positions in the data.

Second, we need to respect the quadratic constraints between matrices, such as $M = CC^\top$ and $N = DD^\top$. Unfortunately, these are non-convex. However, using the same approach as (Xu and Schuurmans, 2005) we can relax these to the convex one-sided constraints

$$M \succeq CC^\top, N \succeq DD^\top, \mathrm{diag}(M) = \mathbf{e}, \mathrm{diag}(N) = \mathbf{e} \tag{5.26}$$

We also need to relax the quadratic constraints that relate M and N (the equivalence relations on singleton and pairwise assignments):

$$N_{ikk-1,j\ell\ell-1} = M_{ik,j\ell} M_{ik-1,j\ell-1} \quad \forall ijk\ell.$$

These are also non-convex, but can be approximated by linear constraints

$$\begin{array}{rcl} N_{ikk-1,j\ell\ell-1} & \leq & M_{ik,j\ell} \\ N_{ikk-1,j\ell\ell-1} & \leq & M_{ik-1,j\ell-1} \\ N_{ikk-1,j\ell\ell-1} & \geq & M_{ik,j\ell} + M_{ik-1,j\ell-1} - 1 \end{array} \tag{5.27}$$

Finally, to pose the training problem, we need to relax the $\{0,1\}$ integer constraints on the matrix entries to $[0,1]$. Putting these pieces together we get a relaxed training criterion for structured predictors that is entirely convex

$$\min_{M,N,C,D\geq 0} \omega(M,N,C,D) \quad \text{subject to } (5.25), (5.26), (5.27)$$

To solve this training problem in the same manner as above, one would then have to re-express this convex problem as a semidefinite program. Unfortunately, we find that the semidefinite program that results is too large to be practical. Therefore, our initial attempt at optimizing this training criterion has taken a different approach: We can obtain a more compact training technique by using a constraint generation method

$$\min_{M,N,C,D,\zeta} \zeta \quad \text{subject to} \quad \zeta \geq \omega(M,N,C,D;\boldsymbol{\mu}_c,\boldsymbol{\nu}_c),\ \forall c \in \mathcal{C} \tag{5.28}$$

Here we keep a finite set of constraints $\mathcal{C}$, where each $\boldsymbol{\mu}_c, \boldsymbol{\nu}_c$ corresponds to a set of dual parameters for an M^{3}N. Then given a current, M,N,C,D, an M^{3}N training algorithm (QP) is used to maximize $\omega(M,N,C,D;\boldsymbol{\mu},\boldsymbol{\nu})$ as a function of μ and ν; hence adding a new constraint to (5.28). Then (5.28) can be solved for a new M,N,C,D by a smaller semidefinite program. By convexity, a fixed point must yield a global solution to the convex problem (5.28). Unfortunately, this training algorithm is still quite expensive. Therefore, in Section 5.4 below I propose some principled alternatives that are much faster, but no longer guaranteed to find a global solution.

5.3 Experimental Evaluation of Global Techniques

To test the quality of the proposed unsupervised training criterion, I implemented the training technique proposed above, using CPLEX for constraint generation and SDPT3 for the outer semidefinite optimizations. The goal in this section is not to assert that I have a practical technology, yet, that one can easily apply to real problems immediately. However, I believe the fundamental idea is important and I first want to demonstrate that the principle works, regardless of computational cost. Then the question of computational efficiency will be revisited and some faster but approximate alternatives will be proposed below.

Data Set:	**CDHMM**	EM
Synth. Data1 (95%)	**3.38 ±0.75**	15.09 ±1.92
Synth. Data2 (90%)	**8.12 ±1.57**	17.49 ±1.81
Synth. Data3 (80%)	**22.12 ±1.40**	30.06 ±1.24
Synth. Data4 (70%)	**31.50 ±1.46**	39.90 ±0.86
Protein Data1	**51.75 ±1.80**	58.11 ±0.47
Protein Data2	**50.38 ±2.04**	57.23 ±0.39

Table 5.1: Prediction error with different methods. Results are averaged over 10 repeats, for each repeat, EM is run 10 times with different starts.

Since we were initially limited in the sizes of the problems I could consider, six small data sets were investigated: four synthetic data sets generated from a 2-state HMM (see Figure 2.4), and two reduced versions of a real protein secondary structure data set obtained from the UCI repository (protein-secondary-structure). In each case, I gathered a sample of labeled sequences, removed the labels, trained the unsupervised HMM models, and used these to relabel the sequences using Viterbi decoding. The accuracy was measured by first optimizing the map between predicted and possible state labels for each method, and then counting the number of misclassified positions in all training sequences. I compared the performance of the proposed convex discriminative HMM training (CDHMM) to standard EM training (EMHMM). The regularization parameter β for CDHMM was set to 0.01 for the synthetic experiments and 1.0 for the protein experiments. EM was run from a random set of initial parameters 10 times. Smoothing had little noticeable effect on EM.

The synthetic data sets were generated from a simple 2-state HMM, where each state emits either 0 or 1 according to emission probability. Emission noise was set equal to the probability of transitioning to the other state. In synthetic data set 1, there is a 5% chance of staying in the same state and a 95% chance of moving to another state; similarly, in synthetic data set 2, 3, and 4 there is a 10%, 20%, and 30% chance respectively of staying in the same state. Thus, the synthetic data sets are incrementally noisier from synthetic data set 1 to 4. To generate training data, we sampled 10 sequences of length 8 from the 2-state HMM.

For the protein sequence experiments, I created two small data sets of 10 subsequences of length 10, with endpoints randomly selected in the data. In the first experiment (protein data 1) a simple HMM was used (Figure 2.4), where y_k is the secondary structure tag (one of 3 values) and x_k is the amino acid tag (one of 20 values). In the second experiment (protein data 2), each observation x_k was set to a window of 3 adjacent amino acid tags (one of 20^3 values).

Table 5.1 shows the classification accuracies achieved by CDHMM and EMHMM on these problems. Here we see that CDHMM learns far more accurate prediction models than EMHMM. In fact, these results are quite strong, supporting the contention that the discriminative training criterion, based on M^{3}Ns, might provide a fundamental improvement over EM for learning structured predictors. Of course, one source of the advantage might simply be that the convex training criterion avoids getting stuck in local minima. However, independent of local minima, we still argue that even in the unsupervised setting, optimizing a discriminative criterion that focuses on $p(\mathbf{y}|\mathbf{x})$ is superior to optimizing a criterion that focuses solely on improving the model of $p(\mathbf{x})$ (which in fact is what EM is trying to do in this case). Below further evidence is presented to attempt to support the second contention.

Unfortunately, the computational cost of the convex training is quite high (hours versus seconds) and I do not yet have a efficient optimization strategy that is able to guarantee global minimization, even though there are no local minima. To try and address this issue, I attempt to formulate more computationally attractive versions of the proposed training criterion.

5.4 Efficient Approximation Techniques

The main idea, currently, for a computationally efficient approximate training method is based on an equivalent reformulation of the training objective. The reformulation we propose sacrifices convexity, but permits more efficient local optimization.

The easiest way to illustrate the idea is to consider the simple 2-class univariate case with the optimization problems:

$$\omega(\mathbf{y}) = \min_{\mathbf{w}} \frac{\beta}{2}\|\mathbf{w}\|^2 + \sum_i [1 - y^i \mathbf{w} \cdot \boldsymbol{\phi}(\mathbf{x}^i)]_+ \tag{5.29}$$

$$= \max_{o \leq \boldsymbol{\lambda} \leq 1} \boldsymbol{\lambda}^\top \mathbf{e} - \frac{1}{2\beta}\langle K \circ \boldsymbol{\lambda}\boldsymbol{\lambda}^\top, \mathbf{y}\mathbf{y}^\top \rangle \tag{5.30}$$

Equations (5.29) and (5.30) give two equivalent expressions for the margin loss as a function of the labeling $\mathbf{y}$, expressed as primal and dual quadratic programs. It is well-known that the primal and the dual solutions are related by $\mathbf{w} = \frac{1}{\beta}\sum_j \lambda_j y^j \boldsymbol{\phi}(\mathbf{x}^j)$. This relationship is, in fact, not accidental, and one can establish several alternative quadratic programs that give the same solution as the standard primal and dual forms.

Proposition 1 *The margin loss, (5.29) and (5.30), is equivalent to*

$$\begin{aligned} \omega(M) &= \min_{0 \leq \boldsymbol{\lambda} \leq 1, \boldsymbol{\xi} \geq 0} \frac{1}{2\beta}\langle K \circ \boldsymbol{\lambda}\boldsymbol{\lambda}^\top, M \rangle + \boldsymbol{\xi}^\top \mathbf{e} \\ &\textit{subject to} \quad \xi_i \geq 1 - \frac{1}{\beta}\sum_j M_{ij}\lambda_j K(\mathbf{x}^i, \mathbf{x}^j) \ \ \forall i \\ &\textit{where} \quad M = \mathbf{y}\mathbf{y}^\top \end{aligned}$$

Proof It is easy to see that the margin loss (5.29) is equivalent to

$$\omega(\mathbf{y}) = \min_{\mathbf{w}, \boldsymbol{\xi}} \frac{\beta}{2}\|\mathbf{w}\|^2 + \sum_{i=1} \xi_i \quad \text{subject to} \quad \begin{array}{l} y^i \mathbf{w} \cdot \phi(\mathbf{x}^i) \geq 1 - \xi_i \ \ \forall i \\ \xi_i \geq 0 \ \ \forall i \end{array} \tag{5.31}$$

Therefore one can formulate the Lagrangian function by introducing a variable for each constraint:

$$L = \frac{\beta}{2}\|\mathbf{w}\|^2 + \sum_{i=1} \xi_i - \sum_{i=1} \lambda_i [y^i \phi(\mathbf{x}^i) \cdot \mathbf{w} - (1 - \xi_i)] - \sum_{i=1} \mu_i \xi_i \quad \boldsymbol{\lambda}, \boldsymbol{\mu} \geq 0 \tag{5.32}$$

By minimizing over $\mathbf{w}$, and $\boldsymbol{\xi}$, setting the respective derivatives to zero, we can obtain

$$\mathbf{w} = \frac{1}{\beta}\sum_{i=1} \lambda_i y^i \phi(\mathbf{x}^i) \tag{5.33}$$

$$\boldsymbol{\lambda} = \mathbf{e} - \boldsymbol{\mu} \tag{5.34}$$

together with all the non-negative constraints $\boldsymbol{\lambda}, \boldsymbol{\mu}, \boldsymbol{\xi} \geq 0$. If we substitute (5.33) into (5.31) and adding the constraints on dual variables, it is not difficult to obtain the following

$$\begin{aligned} \omega(M) &= \min_{0 \leq \boldsymbol{\lambda} \leq 1, \boldsymbol{\xi} \geq 0} \frac{1}{2\beta} \langle K \circ \boldsymbol{\lambda}\boldsymbol{\lambda}^\top, M \rangle + \boldsymbol{\xi}^\top \mathbf{e} \\ & \quad \text{subject to} \quad \xi_i \geq 1 - \frac{1}{\beta} \sum_j M_{ij} \lambda_j K(\mathbf{x}^i, \mathbf{x}^j) \ \ \forall i \\ & \quad \text{where} \quad M = \mathbf{y}\mathbf{y}^\top \end{aligned}$$

∎

That is, the normal *maximization* over the dual variables can be replaced by an equivalent *minimization* over the dual and slack variables. This allows one to re-express the problem of maximizing margin loss as a joint *minimization* between the dual variables $\boldsymbol{\lambda}$ and the equivalence relation M as

$$\min_M \min_{0 \leq \boldsymbol{\lambda} \leq 1, \boldsymbol{\xi} \geq 0} \omega(M; \boldsymbol{\lambda}, \boldsymbol{\xi}) = \boldsymbol{\lambda}^\top (K \circ M) \boldsymbol{\lambda} / 2\beta + \boldsymbol{\xi}^\top \mathbf{e}$$
$$\text{subject to convex constraints}$$

Unfortunately, $\omega(M; \boldsymbol{\lambda}, \boldsymbol{\xi})$ is not jointly convex in M and $\boldsymbol{\lambda}$, meaning that global minimization cannot be easily guaranteed. Nevertheless, the objective is marginally convex in M and $\boldsymbol{\lambda}$, $\boldsymbol{\xi}$. This suggests an alternating minimization approach—first solve a semidefinite program in M given $\boldsymbol{\lambda}$ and $\boldsymbol{\xi}$, then solve a quadratic program in $\boldsymbol{\lambda}$, $\boldsymbol{\xi}$ given M, and so on. A key fact about alternating minimization is that it must make monotonic progress in the objective ω. Given that the loss ω is bounded below, such a procedure is guaranteed to converge to a local minimum.

Although alternating minimization yields superior intermediate solutions to the constraint generation method used above, it still involves a semidefinite program at each alternation, making it still too expensive for large practical problems. However, the key observation is that one can now in fact sidestep semidefinite programming entirely.

Proposition 2 *Given an SVM solution specified by a fixed $\boldsymbol{\lambda}$, the labeling $\mathbf{y}$ (and its equivalence relation $M = \mathbf{y}\mathbf{y}^\top$) that minimizes margin loss is given by the labeling that is consistent with the SVM's predictions.*

Proof In fact, this proposition is fairly obvious: the best labeling for a fixed SVM, in terms of minimizing margin loss, is simply the SVM's predictions. ∎

Even though this proposition is straightforward, one can obtain a very fast approximate training procedure as a result: initialize the labeling, train an SVM, relabel according to the SVM's discriminant, re-train the SVM, and so on. Although this sounds like a naive heuristic, it is in fact a principled coordinate descent method: each iteration, either re-training or re-labeling, is guaranteed to be non-increasing in the margin loss. Thus the alternation must make monotonic progress in the objective and cannot oscillate, except at a fixed point, which corresponds to a local minimum. I have experimented with this approach below and found that it requires a nontrivial number of iterations (typically more than 1) to reach a fixed point—so the procedure is not completely vacuous as one might fear—but on the other hand the number of iterations rarely exceeds 5-10, so the training time is not significantly worse than training a supervised M^3N. Quite obviously, however, this heuristic only finds local minima and will be dependent on good initialization. Surprisingly, however, I have found that this heuristic alternation technique can achieve good results when applied to large scale sequence data, and is still able to surpass EM in the quality of the structured predictors it learns from unlabeled data.

5.5 Experimental Evaluation of Approximation Techniques

In order to gauge the impact of the approximation, the alternating heuristic (ACDHMM) was run on the same small data sets as the constraint generation method. Note that ACDHMM needs some initial labeling, so it was seeded with the Viterbi labeling from a model learned on a single run of EM. The results shown in Table 5.2 show that ACDHMM is not as accurate as the exact CDHMM procedure, but generally offers better results than EM, especially with the complex model (PROT2). However, the main point here is that ACDHMM scales better than CDHMM so it is possible to present results on much larger data sets. To demonstrate this, I use the same protein secondary structure data set but now use complete sequences (from the available set of 110) instead of sampling short

Data Set:	**CDHMM**	**ACDHMM**	EM
Synth. Data1 (95%)	**3.38 ±0.75**	**14.46 ±1.78**	15.09 ±1.92
Synth. Data2 (90%)	**8.12 ±1.57**	**17.34 ±1.52**	17.49 ±1.81
Synth. Data3 (80%)	**22.12 ±1.40**	**26.56 ±1.06**	30.06 ±1.24
Synth. Data4 (70%)	**31.50 ±1.46**	**38.58 ±0.96**	39.90 ±0.86
Protein Data1	**51.75 ±1.80**	**56.67 ±0.47**	58.11 ±0.47
Protein Data2	**50.38 ±2.04**	**53.65 ±0.57**	57.23 ±0.39

Table 5.2: Prediction error including alternating method.

Data Set:	**ACDHMM**	EM
20×2-seq	**43.12 ±2.20**	46.27 ±1.51
10×5-seq	**44.33 ±2.30**	48.67 ±1.51
5×10-seq	**46.44 ±2.12**	48.67 ±1.82

Table 5.3: Prediction error for larger data sets.

segments. Results are shown in Table 5.3 for 20 samples of 2 sequences (20×2-SEQ), 10 samples of 5 sequences (10×5-SEQ), and 5 samples of 10 sequences (5×10-SEQ) taken randomly from the data set. These data sets are much larger than the earlier examples, having, on average, 337, 628, and 1214 structure observations respectively. In all cases, the observations x_k were set to a window of 7 adjacent amino acids. The results show an improvement over EM in a more realistic context.

5.6 Conclusion

I have presented a new discriminative approach to the unsupervised training of hidden Markov models. This technique combines current ideas in discriminative sequence prediction with those in discriminative unsupervised training. To the best of my knowledge this is the first technique to formulate a convex criterion for discriminative unsupervised training. It preserves the advantages of discriminative training criterion, moreover, it gets around the problem of local minimum that the EM algorithm incurs.

The experimental results mirror the experience in supervised learning that, from the perspective of learning a decoder $p(\mathbf{y}|\mathbf{x})$, it is better to use a discriminative training criterion than a joint criterion. I can obtain an exact but expensive training method for this criterion, or fast but inexact training methods, but cannot yet attain both. However, further progress can be achieved along either direction by improving the semidefinte solvers or by developing better approximations, to improve the scalability of the algorithm.

Chapter 6

Robust Supervised Support Vector Machines

In this chapter, I shift attention away from deriving unsupervised and semi-supervised variants of SVMs to addressing another shortcoming SVMs exhibit for *supervised* learning. The fundamental principle of large margin training, though simple and intuitive, has proved to be one of the most effective estimation techniques devised for supervised classification learning problems. The simplest version of the idea is to find a hyperplane that correctly separates binary labeled training data with the largest margin, intuitively yielding maximal robustness to perturbation and reducing the risks of future misclassifications.

Unfortunately, the naive maximum margin principle yields poor results on non-linearly separable data because the solution hyperplane becomes determined by the most misclassified points, causing a breakdown in theoretical and practical performance. In practice, some sort of mechanism is required to prevent training from fixating solely on anomalous data. For the most part, the field appears to have fixated on the soft margin SVM approach to this problem (Cortes and Vapnik, 1995), where one minimizes a combination of the inverse squared margin and linear margin violation penalty (hinge loss).

Unfortunately, the soft margin SVM has serious shortcomings. One drawback is the lack of a probabilistic interpretation of the margin loss, which creates an unintuitive parameter to tune and causes difficulty in modeling overlapping distributions. However, the central drawback I address in this chapter is that outlier points are guaranteed to play a maximal role in determining the decision hyperplane, since they tend to have the largest margin loss and therefore significantly influence the objective. Such dependence is exactly the opposite of what we would like to do with outliers. In this chapter, I augment the standard soft margin SVM scheme with an explicit outlier suppression mechanism.

There have been a few previous attempts to improve the robustness of large margin training to outliers. The theoretical literature has investigated the concept of a robust margin loss that does not increase the penalty after a certain point (Bartlett and Mendelson, 2002; Krause and Singer, 2004; Mason et al., 2000). One problem with these approaches though is that they lose convexity in the training objective, which prevents global optimization. There have also been a few attempts in the literature to propose convex training objectives that can mitigate the effect of outliers. Song et al. (2002) formulate a robust SVM objective by scaling the margin loss by the distance from a class centroid, reducing the losses (hence the influence) of points that lie far from their class centroid. Weston and Herbrich (2000) formulate a new training objective based on minimizing a bound on the leave one out cross validation error of the soft margin SVM. These approaches are discussed in more detail in Section 6.4 below, but one property they share is that they do not attempt to identify outliers, but rather alter the margin loss to reduce the effect of misclassified points. Unfortunately, altering the fundamental margin loss can cause useful theoretical properties to be lost, such as the margin loss being an upper bound on misclassification error, which makes a connection to existing interpretations and generalization analyses much more difficult.

In this chapter I propose a more direct approach to the problem of robust SVM training by formulating outlier detection and removal directly in the standard soft margin framework, without altering the standard hinge loss formulation. I gain several advantages in doing so. First, the robustness of the standard soft margin SVM training is improved by explicit outlier ablation. Second, the approach preserves the standard margin loss and thereby retains a direct connection to standard theoretical analyses of SVMs. Third, one obtains the first practical training algorithm for training on the robust hinge loss proposed in the theoretical literature. Finally, outlier detection itself can be a significant benefit.

Although outlier detection is not the central goal in this chapter, it is an important problem in many areas of machine learning and data mining (Aggarwal and Yu, 2001;

Brodley and Friedl, 1996; Fawcett and Provost, 1997; Tax, 2001; Manevitz and Yoursef, 2001). Most work on outlier detection focuses on the unsupervised case where there is no designated class variable. However, the technique I develop in this chapter is focused on the supervised case, where the class labels are used to help identify outliers in each class.

6.1 Soft Margin SVMs

Here I will focus on the standard soft margin SVM for binary classification. In the primal representation the classifier is given by a linear discriminant on input vectors (2.1), $\hat{y} = \text{sgn}(\mathbf{w} \cdot \mathbf{x})$, parameterized by a weight vector $\mathbf{w}$. Note that I drop the scalar offset b for ease of exposition.

As introduced in Section 2.1.1, given a training set $(\mathbf{x}^1, y^1), ..., (\mathbf{x}^n, y^n)$ represented as an $n \times d$ matrix of (row) feature vectors, X, and a $n \times 1$ vector of training labels, $\mathbf{y} \in \{-1, +1\}^n$, the goal of soft margin SVM training is to minimize a regularized *hinge loss*, which for a training example $(\mathbf{x}^i, y^i)$ is given by:

$$hinge(\mathbf{w}, \mathbf{x}^i, y^i) \;=\; [1 - y^i \mathbf{w} \cdot \mathbf{x}^i]_+$$

Let the misclassification error be denoted by

$$err(\mathbf{w}, \mathbf{x}^i, y^i) \;=\; 1_{(y^i \mathbf{w} \cdot \mathbf{x}^i < 0)}$$

Then it is easy to see that the hinge loss gives an upper bound on the misclassification error.

Proposition 3 $hinge(\mathbf{w}, \mathbf{x}, y) \geq err(\mathbf{w}, \mathbf{x}, y)$

The proof is immediate. See Figure 6.1. The hinge loss is a well motivated proxy for misclassification error, which itself is non-convex and NP-hard to optimize (Kearns et al., 1992; Hoeffgen et al., 1995). Recall that to derive the soft margin SVM, one can derive a quadratic program (2.8), and its dual (2.13) corresponding to minimizing regularized hinge loss (2.7), which I repeat here for convenience

$$\min_{\mathbf{w}} \frac{\beta}{2} \|\mathbf{w}\|^2 + \sum_i [1 - y^i \mathbf{w} \cdot \mathbf{x}^i]_+ \tag{6.1}$$

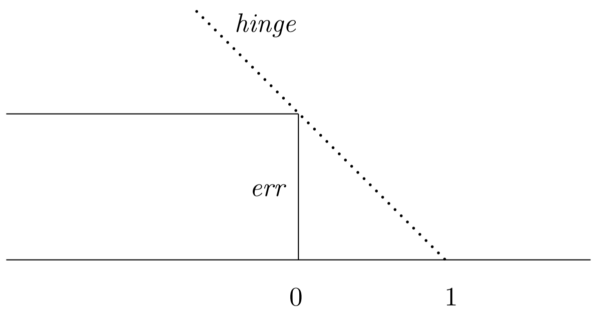


Figure 6.1: Hinge loss vs. misclassification error as a function of $y\mathbf{w}\cdot\mathbf{x}$. The dotted function is the hinge loss (*hinge*), and the step function is the misclassification error (*err*).

$$= \min_{\mathbf{w}} \frac{\beta}{2}\|\mathbf{w}\|^2 + \sum_i \xi_i \quad \text{subject to} \quad \xi_i \geq 1 - y^i\mathbf{w}\cdot\mathbf{x}^i,\ \xi_i \geq 0 \quad \forall i \tag{6.2}$$

$$= \max_{\boldsymbol{\lambda}} \boldsymbol{\lambda}^\top\mathbf{e} - \frac{1}{2\beta}\langle XX^\top \circ \boldsymbol{\lambda}\boldsymbol{\lambda}^\top, \mathbf{y}\mathbf{y}^\top\rangle \quad \text{subject to} \quad 0 \leq \boldsymbol{\lambda} \leq 1 \tag{6.3}$$

Recall that in the dual (6.3), the feature vectors only occur as inner products $\mathbf{x}^i \cdot \mathbf{x}^j$ and therefore can be replaced by a kernel operator $K_{ij} = K(\mathbf{x}^i, \mathbf{x}^j)$.

It is instructive to consider how the soft margin solution is affected by the presence of outliers. In general, the soft margin SVM limits the influence of any single training example, since $0 \leq \lambda_i \leq 1$ by (6.3), while the solution classifier is determined by an $\boldsymbol{\lambda}$-weighted linear combination of the training examples and thus the influence of outlier points is bounded. However, the influence of outlier points is not zero in the soft margin SVM. In fact, of all training points, outliers will still retain maximal influence on the solution, since they will normally have the largest hinge loss. This results in the soft margin SVM still being inappropriately drawn toward outlier points, as Figure 6.3 illustrates. Essentially, the soft margin mechanism is a reasonable approach for handling minor amounts of class overlap, but it is not an effective mechanism for discounting outlier points.

6.2 Robust SVM Training

The main idea in this chapter is to augment the soft margin SVM with explicit indicator variables that can remove outliers entirely. The first application of this approach will be to

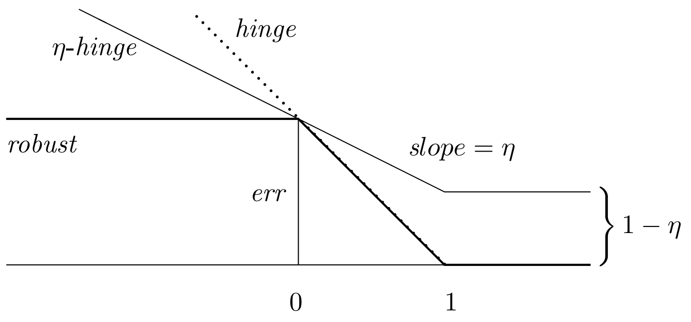


Figure 6.2: Margin losses as a function of $y\mathbf{w}\cdot\mathbf{x}$. The **dotted** function is the hinge loss (*hinge*), the **bold** function is the robust hinge loss (*robust*), the **thin** function is the η-hinge loss (η-*hinge*), and the **step** function is the misclassification error (*err*). Note that η-*hinge* $\geq$ *robust* $\geq$ *err* for $0 \leq \eta \leq 1$. Also *hinge* $\geq$ *robust*. If $y\mathbf{w}\cdot\mathbf{x} \leq 0$, then $\eta = 0$ minimizes η-*hinge*; else $\eta = 1$ minimizes η-*hinge*. Thus $\min_\eta$ η-*hinge* $=$ *robust* for all $y\mathbf{w}\cdot\mathbf{x}$.

show that outlier indicators can be used to directly minimize the robust hinge loss (Bartlett and Mendelson, 2002; Shawe-Taylor and Cristianini, 2004). Then the approach is adapted to focus more specifically on outlier identification.

Define a variable η_i for each training example $(\mathbf{x}^i, y^i)$ such that $0 \leq \eta_i \leq 1$, where $\eta_i = 0$ is intended to indicate that example i is an outlier. Assume initially that these outlier indicators are boolean, $\eta_i \in \{0, 1\}$, and known beforehand. Then one could trivially augment the soft SVM criterion (6.1) to discount the outliers by solving

$$\min_{\mathbf{w}} \frac{\beta}{2}\|\mathbf{w}\|^2 + \sum_i \eta_i[1 - y^i\mathbf{w}\cdot\mathbf{x}^i]_+ \tag{6.4}$$

In this formulation, no loss is charged for any points where $\eta_i = 0$, and these examples are removed from the solution. One problem with this initial formulation, however, is that $\eta_i[1 - y^i\mathbf{w}\cdot\mathbf{x}^i]_+$ is no longer an upper bound on the misclassification error. Therefore, we add a constant term $1 - \eta_i$ to recover an upper bound. Specifically, we define a new loss function

$$\eta\text{-}hinge(\mathbf{w}, \mathbf{x}, y) \quad = \quad \eta\,[1 - y\mathbf{w}\cdot\mathbf{x}]_+ + 1 - \eta$$

With this definition one can show for all $0 \leq \eta \leq 1$

Proposition 4 $\eta\text{-}hinge(\mathbf{w},\mathbf{x},y) \geq err(\mathbf{w},\mathbf{x},y)$

In fact, this upper bound is very easy to establish; see Figure 6.2. Similar to (6.4), minimizing the objective

$$\min_{\mathbf{w}} \frac{\beta}{2}\|\mathbf{w}\|^2 + \sum_i \eta_i\text{-}hinge(\mathbf{w},\mathbf{x}^i,y^i) \tag{6.5}$$

ignores any points with $\eta_i = 0$ since their loss is a constant with respect to $\mathbf{w}$.

Now rather than fix $\boldsymbol{\eta}$ ahead of time, we would like to simultaneously optimize $\boldsymbol{\eta}$ and $\mathbf{w}$, which would achieve concurrent outlier detection and classifier training. To facilitate efficient computation, the outlier indicator variables are relaxed to be $0 \leq \boldsymbol{\eta} \leq 1$. Note that Proposition 4 still applies in this case, and we retain the upper bound on misclassification error for relaxed $\boldsymbol{\eta}$. Thus, we propose the joint training objective

$$\min_{\mathbf{w}} \min_{0\leq\boldsymbol{\eta}\leq 1} \frac{\beta}{2}\|\mathbf{w}\|^2 + \sum_i \eta_i\text{-}hinge(\mathbf{w},\mathbf{x}^i,y^i) \tag{6.6}$$

This objective yields a convex quadratic program in $\mathbf{w}$ given $\boldsymbol{\eta}$, and a linear program in $\boldsymbol{\eta}$ given $\mathbf{w}$. However, (6.6) is not jointly convex in $\mathbf{w}$ and $\boldsymbol{\eta}$, so alternating minimization is not guaranteed to yield a global solution. Instead, I will derive a semidefinite relaxation of the problem that removes all local minima below. However, before deriving a convex relaxation of (6.6) I first establish a very useful and somewhat surprising result: that minimizing (6.6) is equivalent to minimizing the regularized robust hinge loss from the theoretical literature.

6.2.1 Robust Hinge Loss

The robust hinge loss has often been noted as a superior alternative to the standard hinge loss of soft margin SVMs (Krause and Singer, 2004; Mason et al., 2000). This loss is given by

$$robust(\mathbf{w},\mathbf{x},y) \quad = \quad \min(1, hinge(\mathbf{w},\mathbf{x},y))$$

and is illustrated in bold in Figure 6.2.

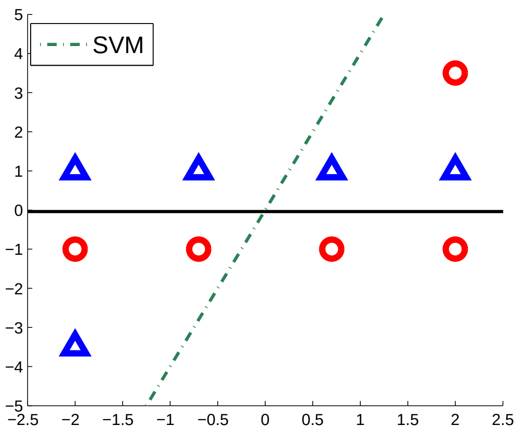


Figure 6.3: Illustrating behavior of SVM given outliers.

The main advantage of robust over regular hinge loss is that the robust loss is bounded, meaning that outlier examples cannot have an effect on the solution beyond that of any other misclassified point. The robust hinge loss also retains an upper bound on the misclassification error, as shown in Figure 6.2. Given such a loss, one can pose the regularized objective

$$\min_{\mathbf{w}} \frac{\beta}{2}\|\mathbf{w}\|^2 + \sum_i \mathit{robust}(\mathbf{w}, \mathbf{x}^i, y^i) \tag{6.7}$$

Unfortunately, even though robust hinge loss has played a significant role in the literature on generalization theory (Bartlett and Mendelson, 2002; Shawe-Taylor and Cristianini, 2004), the minimization objective (6.7) has not been often applied in practice because it is non-convex, and leads to significant difficulties in optimization (Krause and Singer, 2004; Mason et al., 2000).

I can now offer an alternative characterization of robust hinge loss, by showing that it is equivalent to minimizing the η-*hinge* loss introduced earlier. This connection facilitates a new approach to the training problem that I introduce below. First, the η-*hinge* loss can be easily shown to be an upper bound on the robust hinge loss for all η.

Proposition 5 $\eta\text{-}\mathit{hinge}(\mathbf{w}, \mathbf{x}, y) \geq \mathit{robust}(\mathbf{w}, \mathbf{x}, y) \geq \mathit{err}(\mathbf{w}, \mathbf{x}, y)$

Second, minimizing the η-*hinge* loss with respect to η gives the same result as the robust hinge loss

Proposition 6 $\min_{\boldsymbol{\eta}} \eta\text{-}\mathit{hinge}(\mathbf{w}, \mathbf{x}, y) = \mathit{robust}(\mathbf{w}, \mathbf{x}, y)$

Both propositions are straightforward, but can be seen best by examining Figure 6.2. From these two propositions, one can immediately establish the following equivalence.

Theorem 6

$$\begin{aligned} & \min_{\mathbf{w}} \min_{0 \le \boldsymbol{\eta} \le 1} \frac{\beta}{2}\|\mathbf{w}\|^2 + \sum_i \eta_i\text{-}\mathit{hinge}(\mathbf{w}, \mathbf{x}^i, y^i) \\ = & \min_{\mathbf{w}} \frac{\beta}{2}\|\mathbf{w}\|^2 + \sum_i \mathit{robust}(\mathbf{w}, \mathbf{x}^i, y^i) \end{aligned}$$

Moreover, the minimizers are equivalent.

Proof Fix β and define the following:

$$\begin{aligned} f_{\mathit{robust}}(\mathbf{w}) &= \frac{\beta}{2}\|\mathbf{w}\|^2 + \sum_i \mathit{robust}(\mathbf{w}, \mathbf{x}^i, y^i) \\ f_{\mathit{hinge}}(\mathbf{w}, \boldsymbol{\eta}) &= \frac{\beta}{2}\|\mathbf{w}\|^2 + \sum_i \eta_i\text{-}\mathit{hinge}(\mathbf{w}, \mathbf{x}^i, y^i) \\ \mathbf{w}_r &= \arg\min_{\mathbf{w}} f_{\mathit{robust}}(\mathbf{w}) \\ (\mathbf{w}_h, \boldsymbol{\eta}_h) &= \arg\min_{\mathbf{w}} \min_{0 \le \boldsymbol{\eta} \le 1} f_{\mathit{hinge}}(\mathbf{w}, \boldsymbol{\eta}) \\ \eta_r &= \arg\min_{0 \le \boldsymbol{\eta} \le 1} f_{\mathit{hinge}}(\mathbf{w}_r, \boldsymbol{\eta}) \end{aligned}$$

Then from Proposition 5

$$\begin{aligned} \min_{\mathbf{w}} \min_{0 \le \boldsymbol{\eta} \le 1} f_{\mathit{hinge}}(\mathbf{w}, \boldsymbol{\eta}) &= \min_{0 \le \boldsymbol{\eta} \le 1} f_{\mathit{hinge}}(\mathbf{w}_h, \boldsymbol{\eta}) \\ &\ge f_{\mathit{robust}}(\mathbf{w}_h) \ge \min_{\mathbf{w}} f_{\mathit{robust}}(\mathbf{w}) \end{aligned}$$

Conversely, by Proposition 6 we have

$$\begin{aligned} & \min_{\mathbf{w}} f_{\mathit{robust}}(\mathbf{w}) = f_{\mathit{robust}}(\mathbf{w}_r) \\ = & \min_{0 \le \boldsymbol{\eta} \le 1} f_{\mathit{hinge}}(\mathbf{w}_r, \boldsymbol{\eta}) \ge \min_{\mathbf{w}} \min_{0 \le \boldsymbol{\eta} \le 1} f_{\mathit{hinge}}(\mathbf{w}, \boldsymbol{\eta}) \end{aligned}$$

Thus, the two objectives achieve equal values. Finally, the minimizers $\mathbf{w}_r$ and $\mathbf{w}_h$ must be interchangeable, since we have $f_{robust}(\mathbf{w}_r) = f_{hinge}(\mathbf{w}_r, \boldsymbol{\eta}_r) \geq f_{hinge}(\mathbf{w}_h, \boldsymbol{\eta}_h) = f_{robust}(\mathbf{w}_h) \geq f_{robust}(\mathbf{w}_r)$, showing all values are equal. ▮

Therefore minimizing regularized robust loss is equivalent to minimizing the regularized η-*hinge* loss we introduced. Previously, I observed that the regularized η-*hinge* objective can be minimized by alternating minimization on $\mathbf{w}$ and $\boldsymbol{\eta}$. Unfortunately, as Figure 6.2 illustrates, the minimization of $\boldsymbol{\eta}$ given $\mathbf{w}$ always results in boolean solutions that set $\eta_i = 0$ for all misclassified examples and $\eta_i = 1$ for correct examples. Such an approach immediately gets trapped in local minima. Therefore, a better computational approach is required. To develop an efficient training technique for robust loss, a semidefinite relaxation of the problem is derived.

6.2.2 Convex Relaxation

To derive a convex relaxation of (6.6) I need to work in the dual of (6.5). Let $\circ$ denote componentwise multiplication. We then obtain

Proposition 7 *For fixed* $\boldsymbol{\eta}$

$$\begin{aligned}
&\min_{\mathbf{w}} \ \frac{\beta}{2}\|\mathbf{w}\|^2 + \sum_i \eta_i\text{-}\mathit{hinge}(\mathbf{w}, \mathbf{x}^i, y^i) \\
=& \quad \min_{\mathbf{w},\boldsymbol{\xi}} \ \frac{\beta}{2}\|\mathbf{w}\|^2 + \mathbf{e}^\top\boldsymbol{\xi} + \mathbf{e}^\top(\mathbf{e} - \boldsymbol{\eta}) \qquad (6.8)\\
& \qquad\qquad \textit{subject to} \ \ \xi_i \geq 0, \ \ \xi_i \geq \eta_i(1 - y^i\mathbf{w}\cdot\mathbf{x}^i) \ \forall i \\
=& \quad \max_{\boldsymbol{\lambda}} \ \boldsymbol{\eta}^\top(\boldsymbol{\lambda} - \mathbf{e}) - \frac{1}{2\beta}\boldsymbol{\lambda}^\top(XX^\top \circ \mathbf{y}\mathbf{y}^\top \circ \boldsymbol{\eta}\boldsymbol{\eta}^\top)\boldsymbol{\lambda} + n \\
& \qquad\qquad \textit{subject to} \ \ 0 \leq \boldsymbol{\lambda} \leq 1
\end{aligned}$$

Proof The Lagrangian of (6.8) is

$$\begin{aligned}
\mathcal{L}_1 \ &= \frac{\beta}{2}\mathbf{w}^\top\mathbf{w} + \mathbf{e}^\top\boldsymbol{\xi} + \sum \lambda_i(\eta_i - \eta_i y^i\mathbf{w}\cdot\mathbf{x}^i - \xi_i) - \boldsymbol{\nu}^\top\boldsymbol{\xi} + \mathbf{e}^\top(\mathbf{e} - \boldsymbol{\eta}) \\
& \qquad\qquad \text{subject to } \boldsymbol{\lambda} \geq 0, \boldsymbol{\mu} \geq 0
\end{aligned}$$

Computing the gradient with respect to $\boldsymbol{\xi}$ yields $d\mathcal{L}_1/d\boldsymbol{\xi} = \mathbf{e} - \boldsymbol{\lambda} - \boldsymbol{\nu} = 0$, which implies $\boldsymbol{\lambda} \le \mathbf{e}$. The Lagrangian can therefore be equivalently expressed by

$$\begin{aligned} \mathcal{L}_2 &= \frac{\beta}{2}\mathbf{w}^\top\mathbf{w} + \sum \lambda_i(\eta_i - \eta_i y^i \mathbf{w}\cdot\mathbf{x}^i) + \mathbf{e}^\top(\mathbf{e} - \boldsymbol{\eta}) \\ &\qquad \text{subject to } 0 \le \boldsymbol{\lambda} \le 1 \end{aligned}$$

Finally, taking the gradient with respect to $\mathbf{w}$ yields $d\mathcal{L}_2/d\mathbf{w} = \beta\mathbf{w} - \sum \lambda_i\eta_i y^i\mathbf{x}^i = 0$, which implies $\mathbf{w} = \sum \lambda_i\eta_i y^i\mathbf{x}^i/\beta$. Substituting back into $\mathcal{L}_2$ yields the result. ▌

I can subsequently reformulate the joint objective as follows.

Corollary 1

$$\begin{aligned} &\min_{0\le\boldsymbol{\eta}\le 1}\min_{\mathbf{w}} \frac{\beta}{2}\|\mathbf{w}\|^2 + \sum_i \eta_i\text{-}\textit{hinge}(\mathbf{w},\mathbf{x}^i,y^i) \qquad (6.9) \\ &= \min_{0\le\boldsymbol{\eta}\le 1}\max_{0\le\boldsymbol{\lambda}\le 1} \boldsymbol{\eta}^\top(\boldsymbol{\lambda}-\mathbf{e}) - \frac{1}{2\beta}\boldsymbol{\lambda}^\top(XX^\top \circ \mathbf{y}\mathbf{y}^\top \circ \boldsymbol{\eta}\boldsymbol{\eta}^\top)\boldsymbol{\lambda} + n \end{aligned}$$

The significance of this reformulation is that it now allows one to express the inner optimization as a *maximum*, which allows a natural convex relaxation for the outer minimization. The key observation is that $\boldsymbol{\eta}$ appears in the inner maximization only as $\boldsymbol{\eta}$ and the symmetric matrix $\boldsymbol{\eta}\boldsymbol{\eta}^\top$. Using the same idea I exploited in Chapter 3 to 5, if we create a new matrix variable $M = \boldsymbol{\eta}\boldsymbol{\eta}^\top$, we can re-express the problem as a maximum of *linear* functions of $\boldsymbol{\eta}$ and M, yielding a convex objective in $\boldsymbol{\eta}$ and M (Boyd and Vandenberghe, 2004)

$$\min_{0\le\boldsymbol{\eta}\le 1,\, M=\boldsymbol{\eta}\boldsymbol{\eta}^\top}\ \max_{0\le\boldsymbol{\lambda}\le 1} \boldsymbol{\eta}^\top(\boldsymbol{\lambda}-\mathbf{e}) - \frac{1}{2\beta}\boldsymbol{\lambda}^\top(G\circ M)\boldsymbol{\lambda}$$

where $G = XX^\top \circ \mathbf{y}\mathbf{y}^\top$.

The only problem that remains is that $M = \boldsymbol{\eta}\boldsymbol{\eta}^\top$ is a non-convex quadratic constraint. This constraint forces me to make the only approximation: I relax the equality to $M \succeq \boldsymbol{\eta}\boldsymbol{\eta}^\top$ and $\operatorname{diag}(M) = \boldsymbol{\eta}$, which yields a convex optimization problem

$$\begin{aligned} &\min_{0\le\boldsymbol{\eta}\le 1}\min_{M}\max_{0\le\boldsymbol{\lambda}\le 1} \boldsymbol{\eta}^\top(\boldsymbol{\lambda}-\mathbf{e}) - \frac{1}{2\beta}\boldsymbol{\lambda}^\top(G\circ M)\boldsymbol{\lambda} \qquad (6.10) \\ &\qquad \text{subject to } M \succeq \boldsymbol{\eta}\boldsymbol{\eta}^\top,\ \operatorname{diag}(M) = \boldsymbol{\eta} \end{aligned}$$

This problem can be equivalently expressed as a semidefinite program.

Theorem 7 *Solving (6.10) is equivalent to solving the semidefinite program*

$$\min_{\boldsymbol{\eta}, M, \boldsymbol{\mu}, \boldsymbol{\nu}, \zeta} \zeta \quad \textit{subject to } \boldsymbol{\mu} \geq 0,\ \boldsymbol{\nu} \geq 0,\ 0 \leq \boldsymbol{\eta} \leq 1,\ M \succeq \boldsymbol{\eta}\boldsymbol{\eta}^\top,\ \textit{diag(M)=}\boldsymbol{\eta},$$

$$\begin{bmatrix} G \circ M & \boldsymbol{\eta} + \boldsymbol{\mu} - \boldsymbol{\nu} \\ (\boldsymbol{\eta} + \boldsymbol{\mu} - \boldsymbol{\nu})^\top & \frac{2}{\beta}(\zeta - \boldsymbol{\nu}^\top \mathbf{e} + \boldsymbol{\eta}^\top \mathbf{e}) \end{bmatrix} \succeq 0$$

The proof of Theorem 7 is very similar to that of Theorem 1.

This formulation as a semidefinite program admits a polynomial time training algorithm (Nesterov and Nemirovskii, 1994; Boyd and Vandenberghe, 2004). I refer to this algorithm as the robust η-hinge (REH) SVM.

6.3 Explicit Outlier Detection

Note that the technique developed above does not actually identify outliers, but rather just improves robustness against the presence of outliers. That is, a small value of η_i in the computed solution does not necessarily imply that example i is an outlier. To explicitly identify outliers, one needs to be able to distinguish between true outliers and points that are just misclassified because they are in a class overlap region. To adapt the technique to explicitly identify outliers, I propose a minor modification of the previous procedure.

Specifically, let us reconsider a joint optimization of the original objective (6.4), but now add a an explicit constraint that at least a certain proportion ρ of the training examples must not be considered as outliers

$$\min_{\mathbf{w}} \min_{0 \leq \boldsymbol{\eta} \leq 1} \frac{\beta}{2}\|\mathbf{w}\|^2 + \sum_i \eta_i [1 - y^i \mathbf{w} \cdot \mathbf{x}^i]_+ \qquad \text{subject to} \quad \mathbf{e}^\top \boldsymbol{\eta} \geq \rho n \tag{6.11}$$

The difference is that I drop the extra $1 - \eta_i$ term in the η-*hinge* loss and add the proportion ρ constraint. The consequence is that the upper bound on the misclassification error is lost, but the optimization is now free to drop a proportion $1 - \rho$ of the points without penalty, to minimize hinge loss. The points that are dropped should correspond to ones that would have obtained the largest hinge loss; i.e. the outliers. Following the same steps

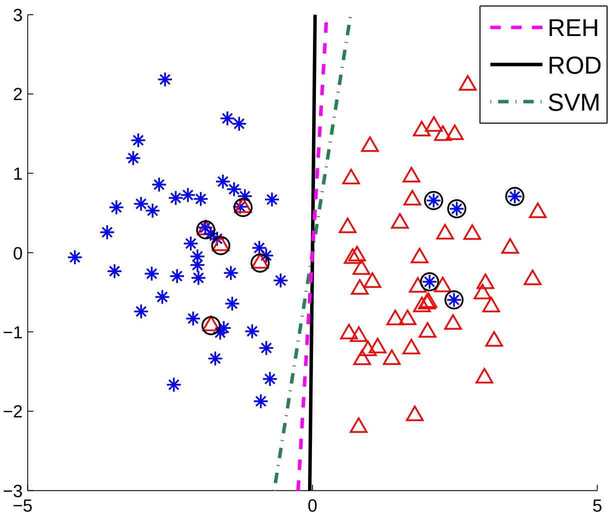


Figure 6.4: Illustrating behavior of REH, ROD and SVM on Gaussian blobs with outliers.

as above, one can derive a dual and then a semidefinite relaxation of this objective (6.11) that allows a reasonable training algorithm:

$$\min_{\boldsymbol{\eta},M,\boldsymbol{\mu},\boldsymbol{\nu},\zeta} \quad \zeta \qquad \text{subject to } \boldsymbol{\mu} \geq 0,\ \boldsymbol{\nu} \geq 0,\ 0 \leq \boldsymbol{\eta} \leq 1,\ M \succeq \boldsymbol{\eta}\boldsymbol{\eta}^\top,\ \mathbf{e}^\top\boldsymbol{\eta} \geq \rho n,$$

$$\begin{bmatrix} G \circ M & \boldsymbol{\eta}+\boldsymbol{\mu}-\boldsymbol{\nu} \\ (\boldsymbol{\eta}+\boldsymbol{\mu}-\boldsymbol{\nu})^\top & \frac{2}{\beta}(\zeta - \boldsymbol{\nu}^\top\mathbf{e}) \end{bmatrix} \succeq 0$$

I refer to this method as the robust outlier detection (ROD) algorithm. Figure 6.4 shows anecdotally that this outlier detection works well in a simple synthetic setting, discovering a much better classifier than the soft margin SVM (hinge loss), while also identifying the outlier points. The robust SVM algorithm developed above also produces good results in this case, but does not identify outliers.

I will examine these two techniques in more detail below, but first I need to discuss competing approaches.

6.4 Comparison to Existing Techniques

Before discussing experimental results, I briefly review related approaches to robust SVM training. Interestingly, the original proposal for soft margin SVMs (Cortes and Vapnik, 1995) already considered alternative losses to hinge loss, based on the transformation $loss(\mathbf{w}, \mathbf{x}, y) = hinge(\mathbf{w}, \mathbf{x}, y)^p$. for $1 \leq p < \infty$. The so called L_2 slack (Lanckriet et al., 2004) is achieved by setting $p = 2$, for example. Unfortunately, transforming the margin penalties by any factor choosing $p > 1$ exaggerates the largest losses and in fact makes the technique more sensitive to outliers than the standard hinge loss. Choosing $p < 1$ improves robustness, but creates a non-convex training problem.

As I mentioned at the start of this chapter, there have been a few more recent attempts to improve the robustness of soft margin SVMs to outliers. The centroid SVM formulation of Song et al. (2002) modifies the margin penalty by shifting the loss according to the distance from the class centroid

$$\min_{\mathbf{w}} \ \frac{1}{2}\|\mathbf{w}\|^2 + \sum_i [1 - y^i \mathbf{w} \cdot \mathbf{x}^i - \lambda \|\mathbf{x}^i - \boldsymbol{\mu}_{y^i}\|^2]_+$$

where $\boldsymbol{\mu}_{y^i}$ is the centroid for class $y^i \in \{-1, +1\}$. Intuitively, examples that are far away from their class centroid will have their margin losses automatically reduced, which diminishes their influence on the solution. If the outliers are indeed far from the class centroid the technique is reasonable; see Figure 6.5. Unfortunately, the motivation is heuristic and loses the upper bound on misclassification error, which blocks any simple theoretical justification.

Another interesting proposal for robust SVM training is the leave-one-out (LOO) SVM and its extension to the adaptive margin SVM (Weston and Herbrich, 2000). The LOO SVM minimizes the leave-one-out error bound on dual soft margin SVMs, derived originally by Jaakkola and Haussler (1999). The bound shows that the misclassification error achieved on a single example i by training a soft margin SVM on the remaining $n - 1$ data points is at most

$$loo\_err(\mathbf{x}^i, y^i) \leq y^i \sum_{j \neq i} \alpha_j y^j \mathbf{x}^i \cdot \mathbf{x}^j$$

where $\boldsymbol{\alpha}$ is the dual solution trained on the entire data set. Weston and Herbrich (2000)

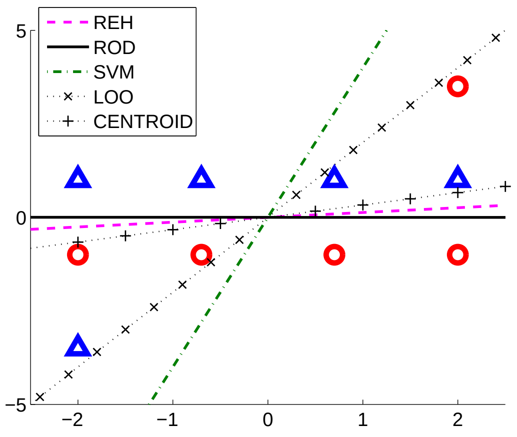


Figure 6.5: Illustrating behavior given outliers.

propose to directly minimize the upper bound on the *loo_err*, leading to

$$\min_{\boldsymbol{\alpha}\geq 0} \sum_i [1 - y^i \mathbf{x}^i \cdot \sum_j \alpha_j y^j \mathbf{x}^j + \alpha_i \|\mathbf{x}^i\|^2]_+ \tag{6.12}$$

Although this objective is hard to interpret as a regularized margin loss, it is closely related to a standard form of soft margin SVM using a modified regularizer

$$\min_{\boldsymbol{\alpha}\geq 0} \sum_i \alpha_i \|\mathbf{x}^i\|^2 + \sum_i [1 - y^i \mathbf{x}^i \cdot \sum_j \alpha_j y^j \mathbf{x}^j + \alpha_i \|\mathbf{x}^i\|^2]_+$$

The objective (6.12) implicitly reduces the influence of outliers, since training examples contribute to the solution only in terms of how well they help predict the labels of *other* training examples. This approach is simple and elegant and only requires linear programming. Nevertheless, its motivation remains a bit heuristic: (6.12) does not give a bound on the leave-one-out error of the LOO SVM technique itself, but rather minimizes a bound on the leave-one-out error of another algorithm (soft margin SVM) that was not actually run on the data. Consequently, the technique is hard to interpret and requires novel theoretical analysis. It can also give anomalous results, as Figure 6.5 indicates. Figure 6.6 shows how these existing techniques compare to my proposed algorithms on the simple synthetic data set.

In contrast to the approach I have proposed above, which is to take a semidefinite relaxation of the non-convex problem to avoid local minima, Collobert et al. (2006) follow

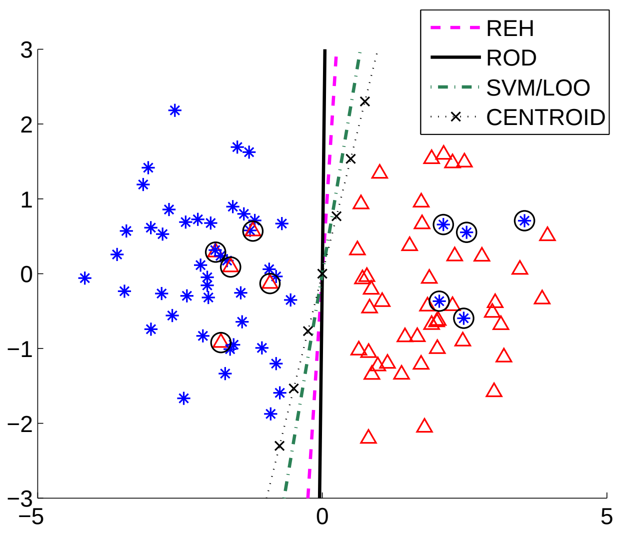


Figure 6.6: Illustrating behavior of different algorithms on Gaussian blobs with outliers.

a different direction, which is to take the non-convex problem as it is, and solve it using some iterative optimization technique. This will not avoid the problem of local minimum, however, better computational efficiency is achieved in return. This work was conducted independently and concurrently with mine (Xu et al., 2006a).

The two algorithms proposed in (Collobert et al., 2006; Xu et al., 2006a) are trying to solve the same problem of minimizing the regularized robust hinge loss (6.7). As discussed previously, this is equivalent to minimizing the regularized η-hinge loss (6.6), via Theorem 6, which I solve by developing a semidefinite relaxation. On the other hand, as the authors of (Collobert et al., 2006) observed, the non-convex part of (6.7), the robust hinge loss $robust(\mathbf{w}, \mathbf{x}^i, y^i)$, can be decomposed into the sum of the convex hinge loss and a concave loss. Thus the problem can be solved using the so called Concave-Convex Procedure (CCCP), which is an iterative algorithm. In each iteration, one only needs to solve a standard SVM, which is a quadratic program. After a finite number of iterations, CCCP is guaranteed to converge. Nevertheless, due to the lack of convexity, only a local minimum will be reached.

In summary, there are two underlying issues related to the problem of minimizing robust hinge loss, which are convexity and scalability. The two algorithms discussed above pursue one of the two directions respectively.

6.5 Experimental Results

I conducted a series of experiments on synthetic and real data sets to compare the robustness of the various SVM training methods, and also to investigate the outlier detection capability of the proposed approach. I implemented the training methods using SDPT3 (Toh et al., 1999) to solve the semidefinite programs proposed in this chapter.

The first experiments were conducted on synthetic data and focused on measuring generalization given outliers, as well as the robustness of the algorithm to their parameters. One Gaussian was assigned per class, with the first given by $\mu = (3, -3)$ and $\Sigma = \begin{pmatrix} 20 & 16 \\ 16 & 20 \end{pmatrix}$ and the second by $-\mu$ and Σ. Since the two Gaussians overlap, the Bayes optimal misclassification error is 2.2%. Outliers were added to the training set by drawing examples uniformly from a ring with inner-radius of R and outer-radius of $R+1$, where R was set to one of the values $15, 35, 55, 75$. These examples were labeled randomly with even probability. In all experiments, the training set contained 50 examples: 20 from each Gaussian and 10 from the ring. The test set contained $1,000$ examples from each class. Here the examples from the ring caused about 10% outliers.

All the experiments were repeated 50 times, drawing a training set and a test set every repetition. All the results reported are averaged over the 50 runs, with a 95% confidence interval. I compared the performance of standard soft margin SVM, robust η-hinge SVM (REH) and the robust outlier detector SVM (ROD). All algorithms were run with the generalization tradeoff parameter set to one of five possible values: $\beta = 10^{-4}, 10^{-2}, 10^{0}, 10^{2}, 10^{4}$. The robust outlier detector SVM was run with outlier parameter set to one of seven possible values, $\rho = 0.2, 0.4, 0.6, 0.8, 0.9, 0.94, 0.98$.

Figure 6.7 shows the results for the two versions of robust hinge loss training versus soft margin SVM, LOO SVM, and centroid SVM. For centroid SVM I used 8 values for the λ parameter and chose the *best* over the test-set. For all other methods I used the best value of β for standard SVM over the test set. The x-axis is the noise level indicated by the radius R and the y-axis is test error.

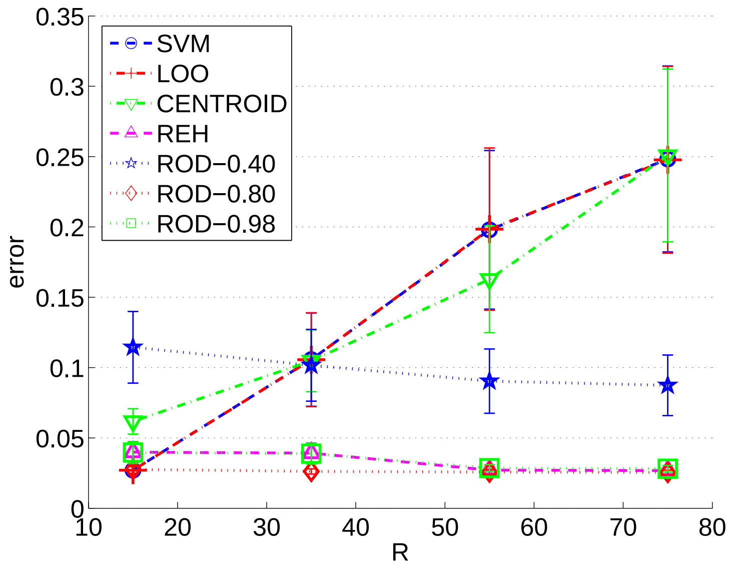


Figure 6.7: Synthetic results: test error as a function of noise level.

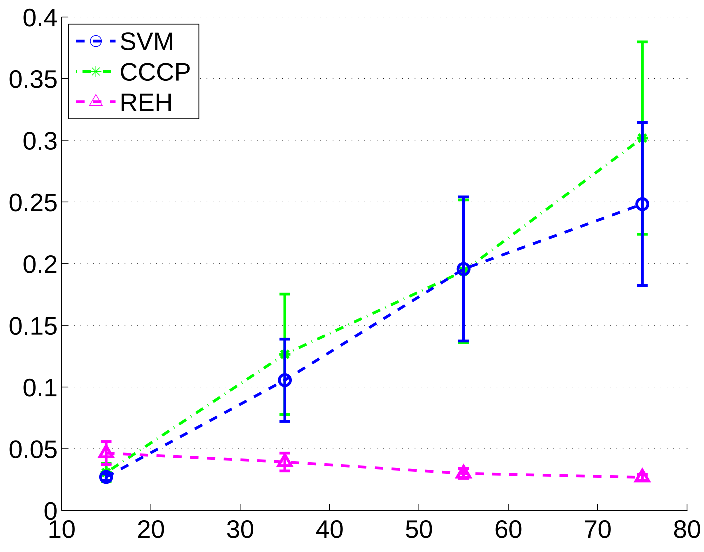


Figure 6.8: Synthetic results: test error as a function of noise level. Comparison of the robust η-hinge algorithm and the CCCP algorithm.

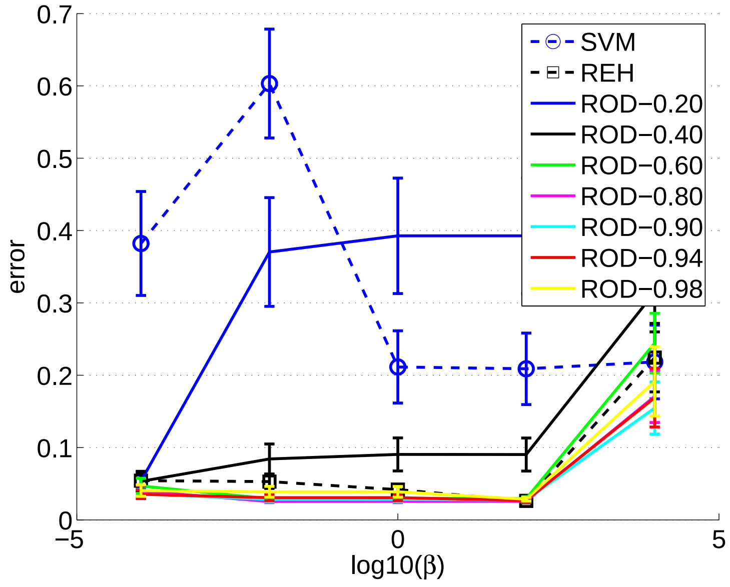


Figure 6.9: Synthetic results: test error as a function of the regularization parameter β.

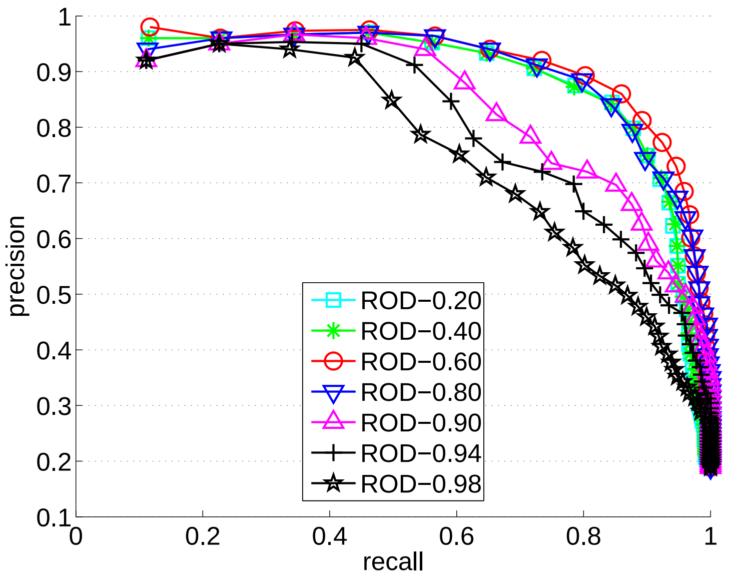


Figure 6.10: Synthetic results: recall-precision curves for the robust outlier detection algorithm (ROD).

These results confirm that the standard soft margin SVM is sensitive to the presence of outliers, as its error rate increases significantly when the radius of the ring increases. By contrast, the robust hinge SVM is not as affected by label noise on distant examples. This is illustrated both in the value of the mean test error and the standard deviation. Here the outlier detection algorithm with $\rho = 0.80$ achieved the best test error, and robust η-hinge SVM second best.

The comparison of the robust η-hinge algorithm and the CCCP approach is clear: the robust η-hinge algorithm I propose in this thesis is capable of finding a global solution in a semidefinite relaxation. However, as stated in previous chapters, despite its polynomial run time, semidefinite programming is still relatively expensive compared to the traditional convex optimization techniques such as LP and QP, leading to difficulty in scaling up to large problems in many real applications. On the other hand, the CCCP approach yields local solutions more efficiently, which makes the larger scale experiments in (Collobert et al., 2006) possible. However, local minima are always an issue, and the quality of solution always depends on the initialization in this approach. This can be clearly illustrated from the comparison shown in Figure 6.8.

Figure 6.9 shows the test error as a function of β for all methods (except the CCCP approach) for high noise level $R = 55$. From the plot a few more conclusions can be drawn: First, when β is close to zero the value of ρ does not affect performance very much. But otherwise, if the value of ρ is too small, then the performance degrades. Third, the robust methods are generally less sensitive to the value of the regularization parameter β. Fourth, if β is very high then it seems that the robust methods converge to the standard SVM.

The outlier detection algorithm was also evaluated as follows. Since the identity of the best linear classifier is known, I identified all misclassified examples and ordered the examples using the η values assigned by the ROD training algorithm. The recall and precision are computed using this ordering and averaged over all 50 runs. Figure 6.10 shows the precision versus recall for the outlier detection algorithm (ROD) for various values of ρ and for the minimal value of β. As we can see from the plot, if ρ is too large (i.e. one guesses that the number of outliers is smaller than their actual number), the detection level is low and the F-measure is about 0.75. For all other values of ρ, I obtain an F-measure of about 0.85.

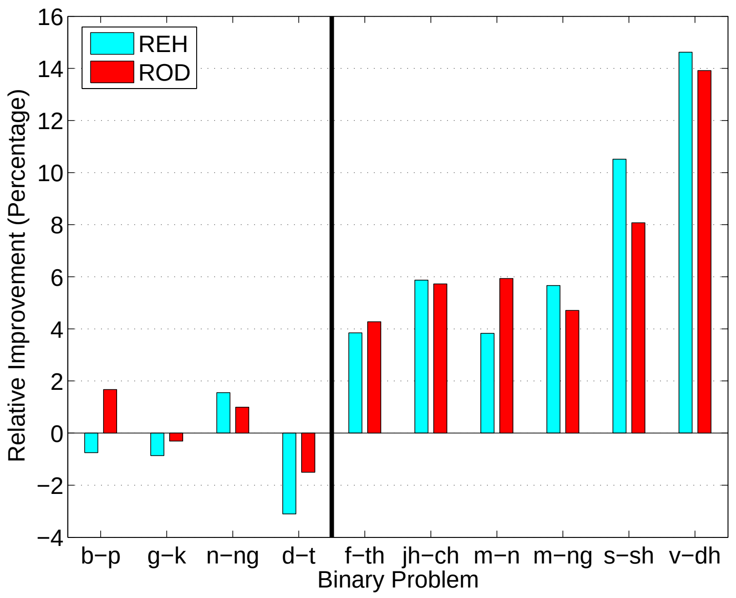


Figure 6.11: Relative improvement of the robust algorithms over standard soft SVM for the speech data.

A few further comments: First, when β is large I can achieve better F-measures by tuning ρ. Second, there are two ways to set ρ. The simple method is to perform cross-validation using the training data and to set ρ to the value that minimized the averaged error. However, an alternative method was found that worked well in practice. If ρ is large, then the graph of sorted η values attains many values near one (corresponding to non-outlier examples) before decreasing to zero for outliers. However, if ρ is small, then all values of η fall below one. Namely, there is a second order phase transition in the maximal value of η, and this phase transition occurs at the value of ρ which corresponds to the true number of outliers.

Given these conclusions, I can proceed with experiments on real data. I conducted experiments on the TIMIT phone classification task. Here I used experimental setup similar to (Gunawardana et al., 2005) and mapped the 61 phonetic labels into 48 classes. 10 pairs of classes were then picked to construct binary classification tasks. I focused mainly on unvoiced phonemes, whose instantiations have many outliers since there is no harmonic underlying source. The ten binary classification problems are identified by a pair of phoneme symbols (one or two Roman letters). For each of the ten pairs 50 random examples were picked from each class, yielding a training set of size 100. Similarly, for test, $2,500$ random examples were picked from each class and generated a test set of size $5,000$. In Preprocessing, mel-frequency cepstral coefficients (MFCCs)[1] were computed with 25ms windows at a 10ms frame rate. The first 13 MFCC coefficients of each frame were retained, along with their first and second time derivatives, and the energy and its first derivative. These coefficient vectors (of dimension 41) were whitened using PCA. A standard representation of speech phonemes is a multivariate Gaussian, which uses the first order and second order interaction between the vector components. Thus each phoneme was represented using a feature vector of dimension 902 using all the first order coefficients (41) and the second order coefficients (861).

For each problem I first ran the soft margin SVM and set β using five-fold-cross validation. This β was then used for all runs of the robust methods. The results, summarized in Figure 6.11, show the relative test error between SVM and the two robust SVM algorithms. Formally, each bar is proportional to $(\epsilon_s - \epsilon_r)/\epsilon_s$, where $\epsilon_s(\epsilon_r)$ is the test error. The results are ordered by their statistical significance for robust hinge SVM (REH) according to McNemar test[2]—from the least significant results (left) to the most significant (right). All the results right to the black vertical line are significant with 95% confidence for *both* algorithms. Here we see that the robust SVM methods achieve significantly better results than the standard SVM in six cases, while the differences are insignificant in four cases. (The difference in performance between the two robust algorithms is not significant.) I also ran the other two methods (LOO SVM and centroid SVM) on this data, and found that they performed worse than the standard SVM in 9 out of 10 cases, and always worse than the two robust SVMs.

[1]Mel-frequency cepstral coefficients are coefficients derived from a type of cepstral representation of the audio clip to represent audio.

[2]The McNemar test is a way to measure the marginal homogeneity of 2×2 contingency tables, introduced by Q. McNemar in 1947. When applied to a 2×2 table which summarizes agreement between two classifiers, the McNemar test illustrates whether or not the difference between the two classifiers is statistically significant.

6.6 Conclusion

In this chapter I proposed a new form of robust SVM training that is based on identifying and eliminating outlier training examples. Interestingly, it was found that this principle provided a new but equivalent formulation to the robust hinge loss often considered in the theoretical literature. The new alternative characterization of robust hinge loss allowed me to derive a practical training procedure for this objective, based on a convex relaxation as a semidefinite program. The resulting training procedure demonstrates superior robustness to outliers than standard soft margin SVM training, and yields generalization improvements in synthetic and real data sets. A useful side benefit of the approach, with some modification, is the ability to explicitly identify outliers as a byproduct of training.

Chapter 7

Conclusion

This thesis presents a novel methodology for applying convex optimization techniques, more specifically semidefinite programming, to various large margin based learning tasks. The new methods share a common framework, with a sequence of steps that can roughly be broken down as: First, augment a standard SVM with indicator variables for unobserved quantities. In unsupervised and semi-supervised SVMs, the new variables are the unobserved y-labels, while in robust SVMs, the new variables are the outlier indicators, neither of which are given during training. The next step is to reformulate the resulting non-convex optimization problems into semidefinite programs via a sequence of convex relaxations. Within this framework, several extensions of SVMs to new scenarios, with promising results, can be developed.

7.1 Summary of Contributions

The first part of the thesis (Chapter 3-4) considers the problem of unsupervised support vector machines (SVMs) for the simple univariate model where one only needs to learn a single label. This can be viewed as a novel alternative to the traditional clustering algorithms such as k-means and spectral clustering. However, the advantage here is a natural connection between supervised learning and unsupervised learning. Through this connection, semi-supervised variants can be easily developed as a direct side benefit.

In the second part (Chapter 5), the methodology is further extended to the multivariate prediction model with structured predictions. The goal here is to train a model with a sequence of labels, where each component not only depends on the input, but also on its neighbors. The work in Chapter 5 is achieved by developing an unsupervised version of maximum margin Markov networks (Taskar et al., 2003), which is a multivariate extension of SVMs.

Instead of unsupervised and semi-supervised SVMs, Chapter 6 deals with the problem of robust SVM training, which is another useful application of semidefinite programming to large margin learning tasks. This work addresses the standard problem of SVM's sensitivity to outliers. Robustness in this case can be achieved by augmenting each example with a variable indicating whether or not it is an outlier, and thus suppressing outliers explicitly. The underlying techniques employed for deriving both unsupervised and robust extensions of SVMs are very much similar.

7.2 Research Directions

There are several specific directions for future research. In particular, there are important questions remaining regarding computation and theoretical analysis.

First, regarding computation, two important issues remain unresolved. One obvious question is proving the hardness of the exact unsupervised SVM problem. Although intuitively there are exponential number of possible label configurations, which makes the problem challenging, a formal proof of the hardness is necessary to illustrate the value of the convex relaxations. Next, even given the polynomial time procedures offered by the semidefinite relaxations, the efficiency of current SDP solvers is still not adequate to scale up to reasonably large problems. This creates a significant problem in practice, because reasonably large data sets are increasingly common in machine learning research. The problem is even more significant in the multivariate case. Thus, developing more efficient training algorithms is a necessary challenge to consider.

Second, regarding the theoretical analysis, there are two main questions to consider. One important question is raised by the fact that the training algorithms I have derived are based on a series of relaxations, and therefore it is important to investigate how these relaxations affect the quality of the final approximation. Another important theoretical issue is to analyze the generalization performance of unsupervised SVMs, and attempt to derive a guarantee that a good labeling that permits a large margin on training data should generalize to new data.

I hope that continued research will provide a theoretically sound and practically efficient framework for the large margin based learning techniques discussed in the thesis, including unsupervised/semi-supervised SVMs and robust SVMs.

Bibliography

C. Aggarwal and P. Yu. Outlier detection for high dimensional data. In *Proceedings of the 2001 ACM SIGMOD International Conference on Management of Data*, 2001.

E. Allwein, R. Shapire, and Y. Singer. Reducing multiclass to binary: a unifying approach for margin classifiers. *Journal of Machine Learning Research*, 1, 2000.

C. Alpert, A. Kahng, and S. Yao. Spectral partitioning: The more eigenvectors, the better. *Discrete Applied Math*, 90, 1999.

Y. Altun, D. McAllester, and M. Belkin. Maximum margin semi-supervised learning for structured variables. In *Advances in Neural Information Processing Systems 18 (NIPS-05)*, 2005.

Y. Altun, I. Tsochantaridis, and T. Hofmann. Hidden markov support vector machines. In *Proceedings International Conference on Machine Learning (ICML-03)*, 2003.

N. Bansal, A. Blum, and S. Chawla. Correlation clustering. In *Conference on Foundations of Computer Science (FOCS-02)*, 2002.

P. Bartlett and S. Mendelson. Rademacher and gaussian complexities: Risk bounds and structural results. *Journal of Machine Learning Research*, 3, 2002.

A. Ben-Hur, D. Horn, H. Siegelman, and V. Vapnik. Support vector clustering. *Journal of Machine Learning Research*, 2, 2001.

K. Bennett and A. Demiriz. Semi-supervised support vector machines. In *Advances in Neural Information Processing Systems 11 (NIPS-98)*, 1998.

T. De Bie and N. Cristianini. Convex methods for transduction. In *Advances in Neural Information Processing Systems 16 (NIPS-03)*, 2003.

B. Boser, I. Guyon, and V. Vapnik. A traininig algorithm for optimal margin classifiers. In *Proceedings of the 5th Annual ACM Workshop on Computational Learning Theory*, 1992.

S. Boyd and L. Vandenberghe. *Convex Optimization.* Cambridge U. Press, 2004.

C. Brodley and M. Friedl. Identifying and eliminating mislabeled training instances. In *Proceedings National Conference on Artificial Intelligence (AAAI-96)*, 1996.

C. Burges. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2(2), 1998.

C. Chennubhotla and A. Jepson. Eigencuts: Half-lives of eigenflows for spectral clustering. In *Advances in Neural Information Processing Systems 15 (NIPS-02)*, 2002.

R. Collobert, F. Sinz, J. Weston, and L. Bottou. Trading convexity for scalability. In *Proceedings International Conference on Machine Learning (ICML-06)*, 2006.

C. Cortes and V. Vapnik. Support-vector networks. *Machine Learning*, 20, 1995.

K. Crammer and Y. Singer. On the algorithmic interpretation of multiclass kernel-based vector machines. *Journal of Machine Learning Research*, 2, 2001.

N. Cristianini and J. Shawe-Taylor. *An Introduction to Support Vector Machines.* Cambridge U. Press, 2000.

T. Dietterich and G. Bakiri. Solving multi-class learning problems via error-correcting ouput codes. *Journal of Artifical Intelligence Research*, 2, 1995.

R. Duda and P. Hart. *Pattern Classification and Scene Analysis.* John Wiley & Sons, Inc, 1973.

B. Durbin, S. Eddy, A. Krogh, and G. Mitchison. *Biological Sequence Analysis.* Cambridge U. Press, 1998.

T. Fawcett and F. Provost. Adaptive fraud detection. *Data Mining and Knowledge Discovery*, 1, 1997.

M. Fiedler. Algebraic connectivity of graphs. *Czechoslovak Mathematics Journal*, 23, 1973.

M. Fiedler. A property of eigenvectors of non-negative symmetric matrices and its application to graph theory. *Czechoslovak Mathematics Journal*, 25, 1975.

M. Goemans and D. Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *Journal of Association for Computing Machinery*, 42, 1995.

G.H. Golub and C.F. Van Loan. *Matrix Computations.* Johns Hopkins University Press, 1996.

A. Gunawardana, M. Mahajan, A. Acero, and J. C. Platt. Hidden conditional random fields for phone classification. In *Proceedings International Conference on Speech Communication and Technology*, 2005.

L. Hagen and A. Kahng. New spectral methods for ratio cut partitioning and clustering. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 11, 1992.

T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning.* Springer, 2001.

C. Helmberg. Semidefinite programming for combinatorial optimization. Technical Report ZIB-Report ZR-00-34, Konrad-Zuse-Zentrum Berlin, 2000.

K. Hoeffgen, K. Van Horn, and U. Simon. Robust trainability of single neurons. *Journal of Computer and System Sciences*, 50(1), 1995.

R. Horn and C. Johnson. *Matrix Analysis.* Cambridge University Press, 1985.

T. Jaakkola and D. Haussler. Probabilistic kernel regression methods. In *Proceedings AISTATS*, 1999.

J.C.Platt. Fast training of support vector machines using sequential minimal optimization. In *Advances in Kernel Methods - Support Vector Learning*. MIT Press, 1998.

F. Jelinek. *Statistical Methods for Speech Recognition*. MIT Press, 1998.

T. Joachims. Transductive inference for text classification using support vector machines. In *Proceedings International Conference on Machine Learning (ICML-99)*, 1999.

D. Jurafsky and J. Martin. *Speech and Language Processing*. Prentice Hall, 2000.

J. Kandola, N. Cristianini, and J. Shaw-Taylor. Spectral kernel methods for clustering. In *Advances in Neural Information Processing Systems 14 (NIPS-01)*, 2001.

M. Kearns, R. Schapire, and L. Sellie. Toward efficient agnostic leaning. In *Proceedings Annual Conference on Learning Theory (COLT-92)*, 1992.

Y. Kluger, R. Basri, J. Chang, and M. Gerstein. Spectral biclustering of microarray cancer data: co-clustering genes and conditions. *Genome Research*, 13, 2003.

N. Krause and Y. Singer. Leveraging the margin more carefully. In *Proceedings International Conference on Machine Learning (ICML-04)*, 2004.

J. Lafferty, A. McCallum, and F. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings International Conference on Machine Learning (ICML-01)*, 2001.

G. Lanckriet, N. Cristianini, P. Bartlett, L. Ghaoui, and M. Jordan. Learning the kernel matrix with semidefinite programming. *Journal of Machine Learning Research*, 5, 2004.

M. Laurent and S. Poljak. On a positive semidefinite relaxation of the cut polytope. *Linear Algebra and its Applications*, 223/224, 1995.

J. MacQueen. Some methods for classification and analysis of multivariate observations. In *Proceedings of the Fifth Berkeley Symposium on Mathematical statistics and probability*, volume 1, 1967.

L. Manevitz and M. Yoursef. One-class svms for document classification. *Journal of Machine Learning Research*, 2, 2001.

L. Mason, J. Baxter, P. Bartlett, and M. Frean. Functional gradient techniques for combining hypotheses. In *Advances in Large Margin Classifiers.* MIT Press, 2000.

R. Neal. Density modeling and clustering using dirichlet diffusion trees. In *Bayesian Statistics 7*, 2003.

Y. Nesterov and A. Nemirovskii. Interior-point polynomial algorithms in convex programming. *SIAM*, 1994.

A. Ng, M. Jordan, and Y. Weiss. On spectral clustering: analysis and an algorithm. In *Advances in Neural Information Processing Systems 14 (NIPS-01)*, 2001.

C. Papadimitriou and K. Steiglitz. *Combinatorial Optimization: Algorithms and Complexity.* Dover Publications, 1998.

L. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. In *Proceedings of the IEEE*, volume 77(2), 1989.

A. Rahimi and B. Recht. Clustering with normalized cuts is clustering with a hyperplane. In *Statistical Learning and Computer Vision*, 2004.

B. Schoelkopf and A. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond.* MIT Press, 2002.

J. Shawe-Taylor, P. Barlett, R. Williamson, and M. Anthony. A framework for structural risk minimization. In *Proceedings of the Ninth Annual Conference on Computational Learning Theory (COLT-96)*, 1996.

J. Shawe-Taylor, P. Barlett, R. Williamson, and M. Anthony. Structural risk minimization over data-dependent hierarchies. *IEEE Trans. on Information Theory*, 44(5), 1998.

J. Shawe-Taylor and N. Cristianini. *Kernel Methods for Pattern Analysis.* Cambridge U. Press, 2004.

J. Shi and J. Malik. Normalized cuts and image segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR-97)*, 1997.

J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8), 2000.

Q. Song, W. Hu, and W. Xie. Robust support vector machine with bullet hole image classification. *IEEE Transaction on Systems, Man and Cybernetics C*, 32(4), 2002.

J. Sturm. Using SeDuMi 1.02, a Matlab toolbox for optimization over symmetric cones. *Optimization Methods and Software*, 11-12, 1999.

C. Swamy. Correlation clustering: maximizing agreements via semidefinite programming. In *Proceedings of the Fifteenth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA-04)*, 2004.

B. Taskar, C. Guestrin, and D. Koller. Max-margin Markov networks. In *Advances in Neural Information Processing Systems 16 (NIPS-03)*, 2003.

D.M.J. Tax. *One-class classification; Concept-learning in the absence of counter-examples.* PhD thesis, Delft University of Technology, 2001.

K. Toh, M. Todd, and R. Tutuncu. SDPT3 – a Matlab software package for semidefinite programming, version 2.1. *Optimization Methods and Software*, 11, 1999.

I. Tsochantaridis, T. Hofmann, T. Joachims, and Y. Altun. Support vector machine learning for interdependent and structured output spaces. In *Proceedings International Conference on Machine Learning (ICML-04)*, 2004.

L. Vandenberghe and S. Boyd. Semidefinite programming. *SIAM Review*, 38(1), 1996.

V. Vapnik. *Estimation of Dependences Based on Empirical Data.* Springer-Verlag, 1982.

V. Vapnik. *The Nature of Statistical Learning Theory.* Springer-Verlag, 1995.

V. Vapnik. *Statistical Learning Theory.* Wiley, 1998.

U. von Luxburg and S. Ben-David. Towards a statistical theory of clustering. In *PASCAL Workshop on Statistics and Optimization of Clustering*, 2005.

U. von Luxburg, O. Bousquet, and M. Belkin. Limits of spectral clustering. In *Advances in Neural Information Processing Systems 17 (NIPS-04)*, 2004.

K. Weinberger, F. Sha, and L. Saul. Learning a kernel matrix for nonlinear dimensionality reduction. In *Proceedings International Conference on Machine Learning (ICML-04)*, 2004.

Y. Weiss. Segmentation using eigenvectors: a unifying view. In *Proceedings International Conference on Computer Vision (ICCV-99)*, 1999.

J. Weston and R. Herbrich. Adaptive margin support vector machines. In *Advances in Large Margin Classifiers*. MIT Press, 2000.

J. Weston and C. Watkins. Support vector machines for multi-class pattern recognition. In *Proceedings of the Seventh European Symposium on Artificial Neural Networks*, 1999.

L. Xu, K. Crammer, and D. Schuurmans. Robust support vector machine training via convex outlier ablation. In *Proceedings National Conference on Artificial Intelligence (AAAI-06)*, 2006a.

L. Xu, J. Neufeld, B. Larson, and D. Schuurmans. Maximum margin clustering. In *Advances in Neural Information Processing Systems 17 (NIPS-04)*, 2004.

L. Xu and D. Schuurmans. Unsupervised and semi-supervised multi-class support vector machines. In *Proceedings National Conference on Artificial Intelligence (AAAI-05)*, 2005.

L. Xu, D. Wilkinson, F. Southey, and D. Schuurmans. Discriminative unsupervised learning of structured predictors. In *Proceedings International Conference on Machine Learning (ICML-06)*, 2006b.