Large Relative Margin and Applications

Pannagadatta K. Shivaswamy

Submitted in partial fulfillment of the
requirements for the degree
of Doctor of Philosophy
in the Graduate School of Arts and Sciences

COLUMBIA UNIVERSITY

2010

ABSTRACT

Large Relative Margin and Applications

Pannagadatta K. Shivaswamy

Over the last decade or so, machine learning algorithms such as support vector machines, boosting *etc.* have become extremely popular. The core idea in these and other related algorithms is the notion of large margin. Simply put, the idea is to geometrically separate two classes with a large separation between them; such a separator is then used to predict the class of unseen test examples. These methods have been extremely successful in practice and have formed a significant portion of machine learning literature. There are several theoretical results which motivate such algorithms. A closer look at such theoretical results reveals that the generalization ability of these methods are strongly linked to the margin as well as some measure of the spread of the data. Yet the algorithms themselves only seem to be maximizing the margin—completely ignoring the spread information. This thesis focuses on addressing this problem; novel formulations, that not only take into consideration the margin but also the spread aspect of the data, are proposed. In particular, relative margin machine, which is a strict generalization of the well known support vector machine is proposed. Further, generalization bounds are derived for the relative margin machines using a novel method of landmark examples. The idea of relative margin is fairly general; its potential is demonstrated by proposing formulations for structured prediction problems as well as for a transductive setup using graph Laplacian. Finally, a boosting algorithm incorporating both the margin information and the spread information is derived as well. The boosting algorithm is motivated from the recent empirical Bernstein bounds. All the proposed variants of the relative margin algorithms are easy to implement, efficiently solvable and typically show significant improvements over their large margin counterparts– on real-world datasets.

Table of Contents

List of Figures

List of Tables

Acknowledgments

The acknowledgments go here.

Dedicated to my late grand parents.

Chapter 1

Introduction

A frequent problem that arises in machine learning is the classification problem. Training examples $(\mathbf{x}_i, \mathbf{y}_i) \in \mathbb{R}^m \times \{\pm 1\}$ are drawn independently and identically distributed (*iid*) from an unknown but a fixed distribution. The aim is to learn a rule $f : \mathbb{R}^m \to \{\pm 1\}$ that can predict well on future examples drawn from the same distribution. A classification algorithm chooses a function from a set of functions typically by optimizing some criterion over the training set. In large margin methods, a function is chosen to maximize the distance between the two classes while minimizing the misclassification rate on the training examples at the same time.

One example of large margin classifiers is the support vector machine (SVM) [Vapnik, 1995; Schölkopf and Smola, 2002; Shawe-Taylor and Cristianini, 2004]. A linear function[1] $f(\mathbf{x}) := \text{sign}(\mathbf{w}^\top \mathbf{x} + b)$ where $\mathbf{w} \in \mathbb{R}^m, b \in \mathbb{R}$ serves as the decision rule through much of this thesis. The parameters of the hyperplane $(\mathbf{w}, b)$ are estimated by maximizing the margin (e.g., the distance between the hyperplanes defined by $\mathbf{w}^\top \mathbf{x} + b = 1$ and $\mathbf{w}^\top \mathbf{x} + b = -1$) while minimizing a weighted upper bound on the misclassification rate on training data (via so-called slack variables); in practice, the margin is maximized by minimizing $\frac{1}{2}\mathbf{w}^\top \mathbf{w}$.

There are various theoretical results that motivate large margin learning in support vector machines. For example, the classic VC-bound [Vapnik, 1995] or the Rademacher bound [Bartlett and Mendelson, 2002; Shawe-Taylor and Cristianini, 2004] *etc.* relate the

[1] In this thesis the dot product $\mathbf{w}^\top \mathbf{x}$ is used with the understanding that it can be replaced with a generalized inner product or by using a kernel for generic objects.

future behavior for all functions in a class to the empirical performance on the data and to so-called measures of complexity. Typical complexity measures that occur in these generalization bounds are the ratio of the radius of the data to the margin, the ratio of the average radius of the training data to the margin and so-forth. Thus, these results show a clear dependence of the generalization error not just on the margin and the performance on the training examples, but also on the spread of the data (such as the radius). Yet, large margin algorithms merely maximize the margin while ignoring the radius information. Further, in the case of support vector machines, the solution can easily be perturbed by an (invertible) affine or scaling transformation of the input space. For instance, by transforming all training and testing inputs by an invertible linear transformation, the SVM solution and its resulting classification performance can be significantly varied. Moreover, this phenomenon is not limited to an explicit adversarial setting, it can naturally occur in many real world classification problems—especially in high dimensions.

This thesis will address such shortcomings in maximum margin solutions which exclusively measure margin disregarding spread information and offer solutions to alleviate such problems. The approaches proposed in this thesis are typically based on controlling the spread while maximizing the margin.

1.1 A motivating example

A two dimensional motivating example will now be shown in detail to highlight the problem with the support vector machines. Consider the simple two dimensional dataset in Figure 1.1 where the goal is to separate the two classes of points: triangles and squares. The figure depicts three scaled versions of the two dimensional problem to illustrate potential problems with the large margin solution.

The three plots on the left show that, as the data is scaled, the maximum margin SVM solution (red or dark shade) deviates from the maximum relative margin solution (green or light shade). Three different scaling scenarios are shown. The three plots on the right show the projections of the examples (that is $\mathbf{w}^\top \mathbf{x} + b$) on the real line for the SVM solution (red or dark shade) and the proposed classifier (green or light shade) under each scaling scenario.

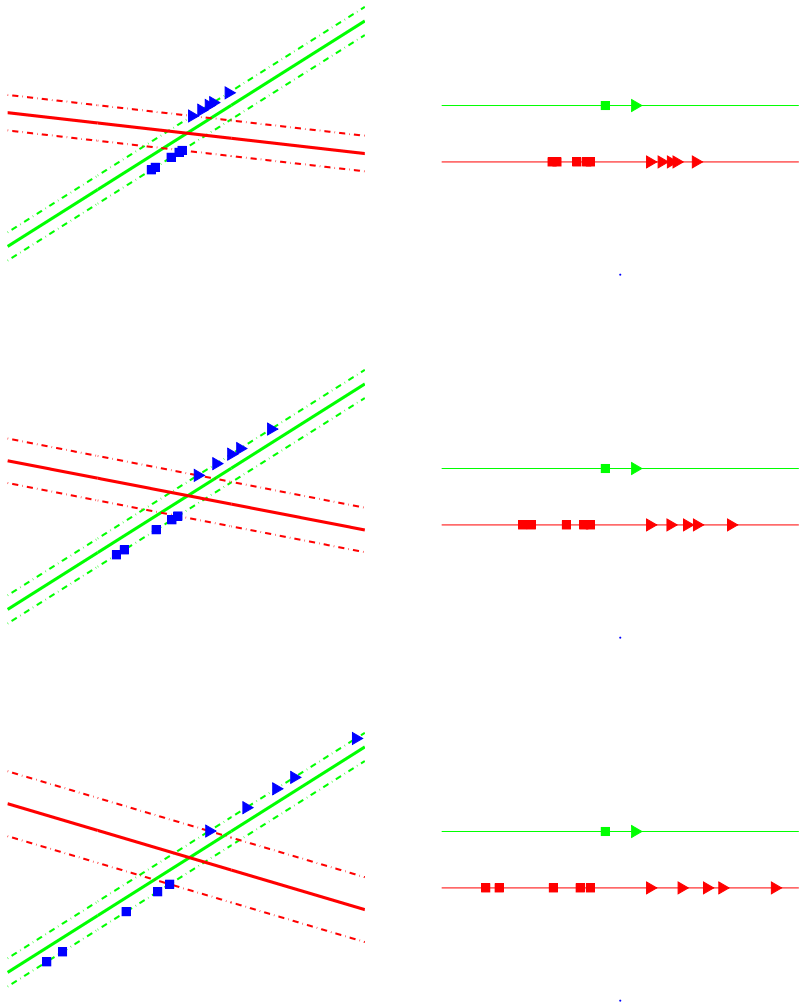

Figure 1.1: A motivating toy example.

These projections have been drawn on separated axes for clarity. The absolute margins for the maximum margin solution (red) are 1.24, 1.51 and 2.08 from top to bottom. For the maximum relative margin solution (green) the absolute margin is merely 0.71. However, the relative margin (the ratio of absolute margin to the spread of the projections) is 41%, 28%, and 21% for the maximum margin solution (red) and 100% for the relative margin solution (green). The scale of all axes is kept locked to permit direct visual comparison. Clearly, the SVM solution achieves the largest margin possible while separating both classes, yet is

this necessarily the best solution?

Consider the plots in the second and the third rows which are obtained merely by scaling the data in the first plot; thus all these three problems correspond to the same discrimination problem up to a scaling factor. With progressive scaling, the SVM increasingly deviates from the maximum relative margin solution (green), clearly indicating that the SVM decision boundary is sensitive to affine transformations of the data. Essentially, the SVM produces a family of different solutions as a result of the scaling. If the SVM solution and its generalization accuracy vary with scaling, an adversary may exploit such scaling to ensure that the SVM performs poorly. Meanwhile, an algorithm producing the maximum relative margin (green) decision boundary could remain resilient to adversarial scaling.

In the toy example, a direction with a small spread in the data produced a good and affine-invariant discriminator which maximized relative margin. Unlike the maximum margin solution, this solution accounts for the spread of the data in various directions. This permits it to recover a solution which has a large margin relative to the spread in that direction. Such a solution would otherwise be overlooked by a maximum margin criterion. A small margin in a correspondingly smaller spread of the data might be better than a large absolute margin with correspondingly larger data spread. This particular weakness in large margin estimation has only received limited attention in previous work.

It is helpful to consider the generative model for the above motivating example. Therein, each class was generated from a one dimensional line distribution with the two classes on two parallel lines. In this case, the maximum relative margin (green) decision boundary should obtain zero test error even if it is estimated from a finite number of examples. However, for finite training data, the SVM solution will make errors and will do so increasingly as the data is scaled further. While it is possible to anticipate these problems and choose kernels or nonlinear mappings to correct them in advance, this is not necessarily practical. The right mapping or kernel is never provided in advance in realistic settings. Instead, one has to estimate kernels and nonlinear mappings, a difficult endeavor which can often exacerbate the learning problem. Similarly, simple data preprocessing (affine whitening to make the dataset zero-mean and unit-covariance or scaling to place the data into a zero-one box) can also fail, possibly because of estimation problems in recovering the correct transformation

(this will be shown in real-world experiments).

The above arguments show that large margin on its own is not enough; it is also necessary to control the spread of the data after projection. Therefore, maximum margin should be traded-off or balanced with the goal of simultaneously minimizing the spread of the projected data, for instance, by bounding the spread $|\mathbf{w}^\top \mathbf{x} + b|$. This will allow the linear classifier to recover large margin solutions not in the absolute sense but rather *relative to* the spread of the data in that projection direction.

1.2 Motivation from an affine invariance perspective

Another motivation for maximum relative margin can be made by reformulating the classification problem altogether. Instead of learning a classifier from data, consider learning an affine transformation on data such that an a priori *fixed* classifier performs well. The data will be mapped by an affine transformation such that it is separated with large margin while it also produces a small radius. Recall that maximum margin classification and SVMs are motivated by generalization bounds based on Vapnik-Chervonenkis complexity arguments. These generalization bounds depend on the ratio of the margin to the radius of the data [Vapnik, 1995]. Similarly, Rademacher generalization bounds [Shawe-Taylor and Cristianini, 2004] also consider the ratio of the trace of the kernel matrix to the margin. Here the radius of the data refers to an R such that $||\mathbf{x}|| \leq R$ for all $\mathbf{x}$ drawn from a distribution.

Instead of learning a classification rule, the optimization problem considered in this section will recover an affine transformation which achieves a large margin from a *fixed* decision rule while also achieving small radius. Assume the classification hyperplane is given a priori via the decision boundary $\mathbf{w}_0^\top \mathbf{x} + b_0 = 0$ with the two supporting margin hyperplanes $\mathbf{w}_0^\top \mathbf{x} + b_0 = \pm\rho$. Here, $\mathbf{w}_0 \in \mathbb{R}^m$ can be an arbitrary unit vector and b_0 is an arbitrary scalar. Consider the problem of mapping all the training points (by an affine transformation $\mathbf{x} \rightarrow \mathbf{A}\mathbf{x} + \mathbf{b}, \mathbf{A} \in \mathbb{R}^{m \times m}, \mathbf{b} \in \mathbb{R}^m$) so that the mapped points (i.e., $\mathbf{A}\mathbf{x}_i + \mathbf{b}$) satisfy the classification constraints $\mathbf{w}_0^\top \mathbf{x} + b_0 = \pm\rho$ while producing small radius, $\sqrt{R}$. The choice of $\mathbf{w}_0$ and b_0 is arbitrary since the affine transformation can completely compensate for the choice. For brevity, denote by $\tilde{\mathbf{A}} = [\mathbf{A}\ \mathbf{b}]$ and $\tilde{\mathbf{x}} = [\mathbf{x}^\top\ 1]^\top$. With this notation, the

affine transformation learning problem is formalized by the following optimization:

$$\begin{aligned} \min_{\tilde{\mathbf{A}},R,\rho} \quad & -\rho + ER \\ & y_i(\mathbf{w}_0^\top \tilde{\mathbf{A}}\tilde{\mathbf{x}}_i + b_0) \geq \rho, && \forall 1 \leq i \leq n \\ & \frac{1}{2}(\tilde{\mathbf{A}}\tilde{\mathbf{x}}_i)^\top(\tilde{\mathbf{A}}\tilde{\mathbf{x}}_i) \leq R && \forall 1 \leq i \leq n. \end{aligned} \tag{1.1}$$

The parameter E trades off between the radius of the affine transformed data and the margin[2] that will be obtained. The following Lemma shows that this affine transformation learning problem is basically equivalent to learning a large margin solution with a small spread.

Lemma 1 *The solution* $\tilde{\mathbf{A}}^*$ *to* (1.1) *is a rank one matrix.*

Proof Consider the Lagrangian of the above problem with Lagrange multipliers $\boldsymbol{\alpha}, \boldsymbol{\lambda}, \geq \mathbf{0}$:

$$\mathcal{L}(\tilde{\mathbf{A}}, \rho, R, \boldsymbol{\alpha}, \boldsymbol{\lambda}) = -\rho + ER - \sum_{i=1}^{n} \alpha_i (y_i(\mathbf{w}_0^\top \tilde{\mathbf{A}}\tilde{\mathbf{x}}_i + b_0) - \rho) + \sum_{i=1}^{n} \lambda_i (\frac{1}{2}(\tilde{\mathbf{A}}\tilde{\mathbf{x}}_i)^\top(\tilde{\mathbf{A}}\tilde{\mathbf{x}}_i) - R).$$

Differentiating the above Lagrangian with respect to $\mathbf{A}$ gives the following expression:

$$\frac{\partial \mathcal{L}(\tilde{\mathbf{A}}, \rho, R, \alpha, \lambda)}{\partial \tilde{\mathbf{A}}} = -\sum_{i=1}^{n} \alpha_i y_i \mathbf{w}_0 \tilde{\mathbf{x}}_i^\top + \tilde{\mathbf{A}} \sum_{i=1}^{n} \lambda_i \tilde{\mathbf{x}}_i \tilde{\mathbf{x}}_i^\top. \tag{1.2}$$

From (1.2), at optimum,

$$\tilde{\mathbf{A}}^* \sum_{i=1}^{n} \lambda_i \tilde{\mathbf{x}}_i \tilde{\mathbf{x}}_i^\top = -\sum_{i=1}^{n} \alpha_i y_i \mathbf{w}_0 \tilde{\mathbf{x}}_i^\top.$$

It is therefore clear that $\tilde{\mathbf{A}}^*$ can always be chosen to have rank one since the right hand side of the expression is just an outer product of two vectors. ■

Lemma 1 gives further intuition on why one should limit the spread of the recovered classifier. Learning a transformation matrix $\tilde{\mathbf{A}}$ so as to maximize the margin while minimizing the radius given an a priori hyperplane $(\mathbf{w}_0, b_0)$ is no different from learning a classification

[2]For brevity, the so-called slack variables have been intentionally omitted since the proof holds in any case.

hyperplane $(\mathbf{w}, b)$ with a large margin as well as a small spread. This is because the rank of the affine transformation $\tilde{\mathbf{A}}^*$ is one; thus, $\tilde{\mathbf{A}}^*$ merely maps all the points $\tilde{\mathbf{x}}_i$ onto a line achieving a certain margin ρ but also limiting the output or spread. This means that finding an affine transformation which achieves a large margin and small radius is equivalent to finding a $\mathbf{w}$ and b with a large margin and with projections constrained to remain close to the origin. Thus, the affine transformation learning problem complements the intuitive arguments in Section 1.1 and also suggests that the learning algorithm should bound the spread of the data.

1.3 Background

Traditionally, controlling spread has been an important theme in classification problems. For instance, classical linear discriminant analysis (LDA) [Duda *et al.*, 2000] finds projections of the data so that the inter-class separation is large while within-class scatter is small. However, the spread (or scatter in this context) is estimated by LDA using only simple first and the second order statistics of the data. While this is appropriate if class-conditional densities are Gaussian, second-order statistics are inappropriate for many real-world datasets and thus, the classification performance of LDA is typically weaker than that of SVMs. The estimation of spread should not make second-order assumptions about the data and should be tied to the margin criterion [Vapnik, 1995]. A similar line of reasoning has been proposed to perform feature selection. [Weston *et al.*, 2000] showed that second order tests and filtering methods on features perform poorly compared to wrapper methods on SVMs which more reliably remove features that have low discriminative value. In this prior work, a feature's contribution to margin is compared to its effect on the radius of the data by computing bounding hyper-spheres rather than simple Gaussian statistics. Unfortunately, there, only axis-aligned feature selection was considered. Similarly, ellipsoidal kernel machines [Shivaswamy and Jebara, 2007] were proposed to normalize data in feature space by estimating bounding hyper-ellipsoids while avoiding inappropriate second-order assumptions. Similarly, the radius-margin bound has been used as a criterion to tune the hyper-parameters of the SVM [Keerthi, 2002]. Another criterion based jointly on ideas from the SVM method as

well as linear discriminant analysis has been studied in [Zhang *et al.*, 2005]. This technique involves first solving the SVM and then solving an LDA problem based on the support vectors that were obtained. While these previous methods showed performance improvements, they relied on multiple-step locally optimal algorithms for interleaving spread information with margin estimation.

A similar method to the relative margin machines was described by [Haffner, 2001], yet that approach started from a different overall motivation. In contrast, this thesis starts with a novel intuition, produces a novel algorithm and provides novel empirical and theoretical support. Another interesting contact point is the second order perceptron framework [Cesa-Bianchi *et al.*, 2002; Cesa-Bianchi *et al.*, 2005] which parallels some of the intuitions underlying the RMM. In an on-line setting, the second order perceptron maintains both a decision rule and a covariance matrix to whiten the data. The mistake bounds it inherits were shown to be better than those of the classical perceptron algorithm. Alternatively, one may consider distributions over classifier solutions which provide a different estimate than the maximum margin setting and have also shown empirical improvements over SVMs [Jaakkola *et al.*, 1999; Herbrich *et al.*, 2001]. In recent papers, [Dredze *et al.*, 2008; Crammer *et al.*, 2009a; Ma *et al.*, 2010] consider a distribution on the perceptron hyperplane. These distribution assumptions permit update rules that resemble a whitening of the data, thus alleviating adversarial affine transformations and producing changes to the basic maximum margin formulation that are similar in spirit to those the RMM provides. In addition, recently, a new batch algorithm called the Gaussian margin machine (GMM) [Crammer *et al.*, 2009b] has been proposed. The GMM maintains a Gaussian distribution over weight vectors for binary classification and seeks the least informative distribution that correctly classifies training data. While the GMM is well motivated from a PAC-Bayesian perspective, the optimization problem itself is expensive involving a log-determinant optimization.

Another alternative route for improving SVM performance includes the use of additional examples. For instance, test samples may be available in semi-supervised or transductive formulations of the SVM [Joachims, 1999; Belkin *et al.*, 2005]. Alternatively, additional data that does not belong to any of the classification classes of interest may be available as in

the so-called Universum approach [Weston *et al.*, 2006; Sinz *et al.*, 2008]. In principle, these methods also change the way margin is measured and the way regularization is applied to the learning problem. There has also been recent work on learning with privileged information [Vapnik and Vashist, 2009] where, in addition to the training examples, extra knowledge known as privileged information is available with each training examlple. However, no privileged information is available at test time. Algorithms that have a faster rate of converegence to Bayes risk are proposed in the above setting. Compared to these, relative margin machine will be proposed for the simple binary classification problem without any additional data. However, it will be extended to transductive learning problems as well.

1.4 Organization

The organization of this thesis is as follows. Based on the motivation and intuition provided in this chapter, several formulations are discussed in Chapter 2. Generalization bounds for the proposed formulations are provided in Chapter 3. The relative margin idea is then extended to structured prediction problem in Chapter 4. In a transductive setup, a large relative margin solution is proposed to optimize the spectrum of a graph Laplacian in Chapter 5. Based on a novel motivation from the recently proposed empirical Bernstein bounds, a new boosting algorithm is proposed in Chapter 6. Finally, conclusions and outlines for possible future work are provided in Chapter 7.

Chapter 2

From absolute margin to relative margin

Based on the motivation provided in the introduction (Chapter 1), several new formulations are discussed in this chapter. This chapter provides an upgrade path from the maximum margin classifier (or SVM) to a maximum relative margin machine (or RMM) formulation.

Given independent identically distributed training examples $(\mathbf{x}_i, y_i)_{i=1}^n$ where $\mathbf{x}_i \in \mathbb{R}^m$ and $y_i \in \{\pm 1\}$ are drawn from $\mathbf{P}(\mathbf{x}, y)$, the support vector machine primal formulation is as follows:

$$\begin{aligned} &\min_{\mathbf{w},b,\xi} \ \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{n}\xi_i && (2.1)\\ &\text{s.t. } y_i(\mathbf{w}^\top \mathbf{x}_i + b) \geq 1 - \xi_i, \quad \xi_i \geq 0 \qquad\qquad \forall 1 \leq i \leq n. \end{aligned}$$

The above is an easily solvable quadratic program (QP) and maximizes the margin by minimizing $\|\mathbf{w}\|^2$. Since real data is seldom separable, slack variables (ξ_i) are used to relax the hard classification constraints. Thus, the above formulation maximizes the margin while minimizing an upper bound on the number of classification errors. The trade-off between the two quantities is controlled by the parameter C. Equivalently, the following dual of the

formulation (2.1) can be solved:

$$\begin{aligned} \max_{\alpha} \ & \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^\top \mathbf{x}_j \\ \text{s.t.} \ & \sum_{i=1}^{n} \alpha_i y_i = 0 \\ & 0 \leq \alpha_i \leq C \qquad \forall 1 \leq i \leq n. \end{aligned} \tag{2.2}$$

Lemma 2 *The formulation in* (2.2) *is invariant to a rotation of the inputs.*

Proof Replace each $\mathbf{x}_i$ with $\mathbf{A}\mathbf{x}_i$ where $\mathbf{A}$ is a rotation matrix such that $\mathbf{A} \in \mathbb{R}^{m \times m}$ and $\mathbf{A}^\top \mathbf{A} = \mathbf{I}$. It is clear that the dual remains the same. ■

However, the dual is not the same if $\mathbf{A}$ is more general than a rotation matrix, for instance, if it is an arbitrary affine transformation.

The above classification framework can also handle non-linear classification readily by making use of Mercer kernels. A kernel function $k : \mathbb{R}^m \times \mathbb{R}^m \rightarrow \mathbb{R}$ replaces the dot products $\mathbf{x}_i^\top \mathbf{x}_j$ in (2.2). The kernel function k is such that $k(\mathbf{x}_i, \mathbf{x}_j) = \langle \phi(\mathbf{x}_i), \phi(\mathbf{x}_j) \rangle$, where $\phi : \mathbb{R}^m \rightarrow \mathcal{H}$ is a mapping to a Hilbert space. Thus, solving the SVM dual formulation (2.2) with a kernel function can give a non-linear solution in the input space. In the rest of this chapter, $\mathbf{K} \in \mathbb{R}^{n \times n}$ denotes the Gram matrix whose individual entries are given by $K_{ij} = k(\mathbf{x}_i, \mathbf{x}_j)$. When applying Lemma 2 on a kernel defined feature space, the affine transformation is on $\phi(\mathbf{x}_i)$ and not on $\mathbf{x}_i$.

Based on the motivation in the introduction, several formulations are discussed in the rest of this chapter. In particular, the ellipsoidal kernel machines (EKM) is discussed in Section 2.1. A formulation based on affine pre-processing of the data is proposed in Section 2.2. The main contributions are in Section 2.3 where the relative margin machine formulation is proposed. Further, efficient ways of solving it and a variant of it are discussed as well. Detailed experimentation on real-world as well as toy datasets are provided in Section 2.4. This chapter ends with a summary in Section 2.5. Generalization considerations of the algorithm are postponed until the next chapter.

2.1 Ellipsoidal kernel machines

The main idea of ellipsoidal kernel machine (abbreviated as EKM) is to consider a family of classifiers with better VC-dimension estimates compared to the support vector machines. The true risk of a classifier can be upper bounded by the empirical risk and terms that involve the VC-dimension of the family. Uniform convergence bounds show the convergence of the empirical risk to the true risk. In other words, with probability at least $1 - \delta$, the following bound holds simultaneously for all $\mathbf{w}$ from a class of VC-dimension h [Vapnik, 1995]:

$$R(\mathbf{w}) \leq \hat{R}(\mathbf{w}) + \sqrt{\frac{h(\log(2n/h) + 1) - \log(\delta/4)}{n}}$$

where, $R(\mathbf{w})$ denotes the true risk of the classifier[1] $\mathbf{w}$ whereas $\hat{R}(\mathbf{w})$ denotes the empirical risk. The above bound suggests low error on the training examples as well as a small VC-dimension for the family of classifiers considered.

One typical assumption on the data in deriving risk bounds for the support vector machines is that they are drawn from a distribution such that the examples are bounded. On bounded data, a large margin classifier gives rise to the so called gap-tolerant classifier [Vapnik, 1995; Burges, 1998]. Typically, the data is assumed to be bounded by a sphere of certain radius (R such that $\|\mathbf{x}\| \leq R$). Thus, finding a large margin solution on bounded data corresponds to the so-called spherical gap tolerant classifiers. Instead of bounding the data by a sphere, [Shivaswamy and Jebara, 2007] considered the tightest bounding ellipsoid (minimum volume) around the data. The VC-dimension of an ellipsoidal gap-tolerant classifier was shown to be smaller compared to the VC-dimension of a spherical gap-tolerant classifier. Based on this motivation, an algorithm to first estimate the tightest bounding ellipsoid and then to estimate the large margin solution considering the shape of the bounding ellipsoid was proposed. The steps of the ellipsoidal kernel machines are outlined in **Algorithm** 2.1. The algorithm essentially finds the tightest bounding ellipsoid (characterized by minimum volume), whitens the data with the shape matrix of the ellipsoid and then finds the support vector machine solution on the whitened data.

[1]The bias term b is ignored here for brevity.

In fact, estimating the bounding ellipsoid around the data can be solved via the following optimization problem:

$$\begin{aligned} &\min_{\mathbf{A},\mathbf{b},\tau} \; -\ln|\mathbf{A}| + E\sum_{i=1}^{n}\tau_i \\ &\text{s.t. } \|\mathbf{A}\mathbf{x}_i - \mathbf{b}\|^2 \leq 1+\tau_i,\ \tau_i \geq 0 \qquad\qquad \forall 1 \leq i \leq n. \end{aligned} \tag{2.3}$$

where, τ_i's are the slack variables to handle the outliers. While the above optimization problem is parametrized by $\mathbf{A}$ and $\mathbf{b}$, the actual parameters of the ellipsoid ($(\mathbf{x}-\boldsymbol{\mu})^\top\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})=1$) are given by

$$\boldsymbol{\Sigma} = \mathbf{A}^{-\frac{1}{2}}, \qquad \mathbf{b} = \boldsymbol{\Sigma}^{-\frac{1}{2}}\boldsymbol{\mu}.$$

Algorithm 2.1 The EKM training algorithm.

Require: Input : $(\mathbf{x}_i, y_i)_{i=1}^n$, Parameters: C, E

1: Solve formulation (2.3) using $(\mathbf{x}_i)_{i=1}^n$ with parameter E.
2: Compute $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ using $\boldsymbol{\Sigma} = \mathbf{A}^{-\frac{1}{2}}$ and $\boldsymbol{\mu} = \boldsymbol{\Sigma}^{\frac{1}{2}}\mathbf{b}$.
3: Transform the data using $\mathbf{u}_i = \boldsymbol{\Sigma}^{-\frac{1}{2}}(\mathbf{x}_i - \boldsymbol{\mu})$ for all i
4: Train SVM formulation (2.1) with $(\mathbf{u}_i, y_i)_{i=1}^n$, parameter C, output $(\mathbf{w}, b)$

Further, it was shown that all the steps of the **Algorithm** 2.1 (including ellipsoid estimation) could be performed in a kernel defined feature space. Although marginal improvements on a number of datasets are observed in [Shivaswamy and Jebara, 2007], the ellipsoid estimation step is computationally very intensive. With n training examples of dimension m, the computational complexity of estimating the tightest bounding ellipsoid is $\mathcal{O}(n^{2.5}m^2)$. Moreover, the algorithm comprises of multiple steps rather than a joint optimization formulation. Thus, EKM is not practical on datasets with more than a few hundred training examples. Thus, ellipsoidal kernel machines are not discussed further in this thesis.

2.2 The whitened SVM

Another way of limiting sensitivity to affine transformations while recovering a large margin solution is to whiten the data with the covariance matrix prior to estimating the SVM

solution. This may also reduce the bias towards regions of large data spread as discussed in the previous chapter. Denote by

$$\boldsymbol{\Sigma} = \frac{1}{n}\sum_{i=1}^{n}\mathbf{x}_i\mathbf{x}_i^\top - \frac{1}{n^2}\sum_{i=1}^{n}\mathbf{x}_i\sum_{j=1}^{n}\mathbf{x}_j^\top, \quad \text{and} \quad \boldsymbol{\mu} = \frac{1}{n}\sum_{i=1}^{n}\mathbf{x}_i,$$

the sample covariance and sample mean, respectively. Now, consider the following formulation called $\boldsymbol{\Sigma}$-SVM:

$$\begin{aligned} \min_{\mathbf{w},b,\xi} \ & \frac{1-D}{2}\|\mathbf{w}\|^2 + \frac{D}{2}\|\boldsymbol{\Sigma}^{\frac{1}{2}}\mathbf{w}\|^2 + C\sum_{i=1}^{n}\xi_i \\ \text{s.t. } & y_i(\mathbf{w}^\top(\mathbf{x}_i - \boldsymbol{\mu}) + b) \geq 1 - \xi_i,\ \xi_i \geq 0 \qquad \forall 1 \leq i \leq n \end{aligned} \tag{2.4}$$

where $0 \leq D \leq 1$ is an additional parameter that trades off between the two regularization terms. When $D = 0$, (2.4) gives back the usual SVM primal (although on translated data). The dual of (2.4) can be shown to be:

$$\begin{aligned} \max_{\alpha} \ & \sum_{i=1}^{n}\alpha_i - \frac{1}{2}\sum_{i=1}^{n}\alpha_i y_i(\mathbf{x}_i - \boldsymbol{\mu})^\top((1-D)\mathbf{I} + D\boldsymbol{\Sigma})^{-1}\sum_{j=1}^{n}\alpha_j y_j(\mathbf{x}_j - \boldsymbol{\mu}) \\ \text{s.t. } & \sum_{i=1}^{n}\alpha_i y_i = 0 \\ & 0 \leq \alpha_i \leq C \qquad \forall 1 \leq i \leq n. \end{aligned} \tag{2.5}$$

It is easy to see that the above formulation (2.5) is translation invariant and tends to an affine invariant solution when D tends to one. However, there are some problems with this formulation. First, the whitening process only considers second order statistics of the input data which may be inappropriate for non-Gaussian datasets. Furthermore, there are computational difficulties associated with whitening. Consider the following term:

$$(\mathbf{x}_i - \boldsymbol{\mu})^\top((1-D)\mathbf{I} + D\boldsymbol{\Sigma})^{-1}(\mathbf{x}_j - \boldsymbol{\mu}).$$

When $0 < D < 1$, it can be shown, by using the Woodbury matrix inversion formula, that the above term can be kernelized as

$$\begin{aligned} \hat{k}(\mathbf{x}_i, \mathbf{x}_j) = & \frac{1}{1-D}\left(k(\mathbf{x}_i, \mathbf{x}_j) - \frac{\mathbf{K}_i^\top\mathbf{1}}{n} - \frac{\mathbf{K}_j^\top\mathbf{1}}{n} + \frac{\mathbf{1}^\top\mathbf{K}\mathbf{1}}{n^2}\right) \\ -\frac{1}{1-D}\left(\left(\mathbf{K}_i - \frac{\mathbf{K}\mathbf{1}}{n}\right)^\top\left(\frac{\mathbf{I}}{n} - \frac{\mathbf{1}\mathbf{1}^\top}{n^2}\right)\left[\frac{1-D}{D}\mathbf{I} + \mathbf{K}\left(\frac{\mathbf{I}}{n} - \frac{\mathbf{1}\mathbf{1}^\top}{n^2}\right)\right]^{-1}\left(\mathbf{K}_j - \frac{\mathbf{K}\mathbf{1}}{n}\right)\right), \end{aligned}$$

where $\mathbf{K}_i$ is the i^{th} column of $\mathbf{K}$. This implies that the $\mathbf{\Sigma}$-SVM can be solved merely by solving (2.2) after replacing the kernel with $\hat{k}(\mathbf{x}_i, \mathbf{x}_j)$ as defined above. Note that the above formula involves a matrix inversion of size n, making the kernel computation alone $\mathcal{O}(n^3)$. Even performing whitening as a preprocessing step in the feature space would involve this matrix inversion which is often computationally much more expensive than obtaining a support vector machine solution.

2.3 Relative margin machines

While both EKM and $\mathbf{\Sigma}$-SVM do address some of the issues of data spread, they are computationally prohibitive. A more direct and efficient approach to control the spread is possible and will be proposed next.

The SVM will be modified such that the projections on the training examples remain bounded. A parameter will also be introduced that helps trade off between large margin and small spread of the projection of the data. This formulation will initially be solved by a quadratically constrained quadratic program (QCQP) in this section. The dual of this formulation will also be of interest and yield further geometric intuitions.

Consider the following formulation called the relative margin machine (RMM):

$$\begin{aligned} \min_{\mathbf{w},b} \ & \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{n}\xi_i & & (2.6) \\ \text{s.t. } & y_i(\mathbf{w}^\top \mathbf{x}_i + b) \geq 1 - \xi_i, \quad \xi_i \geq 0 & & \forall 1 \leq i \leq n \\ & \frac{1}{2}(\mathbf{w}^\top \mathbf{x}_i + b)^2 \leq \frac{B^2}{2} & & \forall 1 \leq i \leq n. \end{aligned}$$

This formulation is similar to the SVM primal (2.1) except for the additional constraints $\frac{1}{2}(\mathbf{w}^\top \mathbf{x}_i + b)^2 \leq \frac{B^2}{2}$. The formulation has one extra parameter B in addition to the SVM parameter C. When B is large enough, the above QCQP gives the same solution as the SVM. Also note that only settings of $B > 1$ are meaningful since a value of B less than one would prevent any training examples from clearing the margin, i.e., none of the examples could satisfy $y_i(\mathbf{w}^\top \mathbf{x}_i + b) \geq 1$ otherwise. Let $\mathbf{w}_C$ and b_C be the solutions obtained by solving the SVM (2.1) for a particular value of C. It is clear, then, that $B > \max_i |\mathbf{w}_C^\top \mathbf{x}_i + b_C|$, makes the constraint on the second line in the formulation (2.6) inactive for each i and the

solution obtained is the same as the SVM estimate. This gives an upper threshold for the parameter B so that the RMM solution is not trivially identical to the SVM solution.

As B is decreased, the RMM solution increasingly differs from the SVM solution. Specifically, with a smaller B, the RMM still finds a large margin solution but with a smaller projection of the training examples. By trying different B values (within the aforementioned thresholds), different large relative margin solutions are explored. It is helpful to next consider the dual of the RMM problem.

The Lagrangian of (2.6) is given by:

$$\begin{aligned}\mathcal{L}(\mathbf{w}, b, \alpha, \lambda, \beta) =& \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{n}\xi_i - \sum_{i=1}^{n}\alpha_i\left(y_i(\mathbf{w}^\top\mathbf{x}_i + b) - 1 + \xi_i\right) - \sum_{i=1}^{n}\beta_i\xi_i \\ &+ \sum_{i=1}^{n}\lambda_i\left(\frac{1}{2}(\mathbf{w}^\top\mathbf{x}_i + b)^2 - \frac{1}{2}B^2\right),\end{aligned}$$

where $\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\lambda} \geq 0$ are the Lagrange multipliers corresponding to the constraints. Differentiating with respect to the primal variables and equating to zero produces:

$$\begin{aligned}&(\mathbf{I} + \sum_{i=1}^{n}\lambda_i\mathbf{x}_i\mathbf{x}_i^\top)\mathbf{w} + b\sum_{i=1}^{n}\lambda_i\mathbf{x}_i = \sum_{i=1}^{n}\alpha_i y_i\mathbf{x}_i, \\ &\frac{1}{\boldsymbol{\lambda}^\top\mathbf{1}}(\sum_{i=1}^{n}\alpha_i y_i - \sum_{i=1}^{n}\lambda_i\mathbf{w}^\top\mathbf{x}_i) = b, \\ &\qquad\qquad \alpha_i + \beta_i = C \qquad\qquad \forall 1 \leq i \leq n.\end{aligned}$$

Denoting by

$$\boldsymbol{\Sigma}_{\boldsymbol{\lambda}} = \sum_{i=1}^{n}\lambda_i\mathbf{x}_i\mathbf{x}_i^\top - \frac{1}{\boldsymbol{\lambda}^\top\mathbf{1}}\sum_{i=1}^{n}\lambda_i\mathbf{x}_i\sum_{j=1}^{n}\lambda_j\mathbf{x}_j^\top, \quad \text{and} \quad \boldsymbol{\mu}_{\boldsymbol{\lambda}} = \frac{1}{\boldsymbol{\lambda}^\top\mathbf{1}}\sum_{i=1}^{n}\lambda_i\mathbf{x}_i,$$

the dual of (2.6) can be shown to be:

$$\max_{\boldsymbol{\alpha},\boldsymbol{\lambda}} \sum_{i=1}^{n}\alpha_i - \frac{1}{2}\sum_{i=1}^{n}\alpha_i y_i(\mathbf{x}_i - \boldsymbol{\mu}_{\boldsymbol{\lambda}})^\top(\mathbf{I} + \boldsymbol{\Sigma}_{\boldsymbol{\lambda}})^{-1}\sum_{j=1}^{n}\alpha_j y_j(\mathbf{x}_j - \boldsymbol{\mu}_{\boldsymbol{\lambda}}) + \frac{1}{2}B^2\sum_{i=1}^{n}\lambda_i \tag{2.7}$$

$$\text{s.t. } 0 \leq \alpha_i \leq C \quad \lambda_i \geq 0 \qquad\qquad \forall 1 \leq i \leq n.$$

Moreover, the optimal $\mathbf{w}$ can be shown to be:

$$\mathbf{w} = (\mathbf{I} + \boldsymbol{\Sigma}_{\boldsymbol{\lambda}})^{-1}\sum_{i=1}^{n}\alpha_i y_i(\mathbf{x}_i - \boldsymbol{\mu}_{\boldsymbol{\lambda}}).$$

Note that the above formulation is translation invariant since $\boldsymbol{\mu_\lambda}$ is subtracted from each $\mathbf{x}_i$. $\boldsymbol{\Sigma_\lambda}$ corresponds to a shape matrix (which is potentially low rank) determined by $\mathbf{x}_i$'s that have non-zero λ_i. From the Karush-Kuhn-Tucker (KKT) conditions of (2.6) it is clear that $\lambda_i(\frac{1}{2}(\mathbf{w}^\top \mathbf{x}_i + b)^2 - \frac{B^2}{2}) = 0$. Consequently $\lambda_i > 0$ implies $(\frac{1}{2}(\mathbf{w}^\top \mathbf{x}_i + b)^2 - \frac{B^2}{2}) = 0$. Notice the similarity in the two dual formulations in (2.5) and (2.7); both formulations look similar except for the choice of $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ which transform the inputs. The RMM in (2.7) whitens data with the matrix $(\mathbf{I} + \boldsymbol{\Sigma_\lambda})$ while simultaneously solving an SVM-like classification problem. While this is similar in spirit to the $\boldsymbol{\Sigma}$-SVM, the matrix $(\mathbf{I} + \boldsymbol{\Sigma_\lambda})$ is being estimated directly to optimize the margin with a small data spread. The $\boldsymbol{\Sigma}$-SVM only whitens data as a preprocessing independently of the margin and the labels. The $\boldsymbol{\Sigma}$-SVM is equivalent to the RMM only in the rare situation when all $\lambda_i = t$ for some t which makes the $\boldsymbol{\mu_\lambda}$ and $\boldsymbol{\Sigma_\lambda}$ in the RMM and $\boldsymbol{\Sigma}$-SVM identical up to a scaling factor.

In practice, the above formulation will not be solved since it is computationally impractical. Solving (2.7) requires semi-definite programming (SDP) which prevents the method from scaling beyond a few hundred data points. Instead, an equivalent optimization will be used which gives the same solution but only requires quadratic programming. This is achieved by simply replacing the constraint $\frac{1}{2}(\mathbf{w}^\top \mathbf{x}_i + b)^2 \leq \frac{1}{2}B^2$ with the two equivalent linear constraints: $(\mathbf{w}^\top \mathbf{x}_i + b) \leq B$ and $-(\mathbf{w}^\top \mathbf{x}_i + b) \leq B$. With these linear constraints replacing the quadratic constraint, the problem is now merely a QP. In the primal, the QP has $4n$ constraints (including $\boldsymbol{\xi} \geq 0$) instead of the $2n$ constraints in the SVM. Thus, the RMM's quadratic program has the same order of complexity as the SVM. In the next section, an efficient implementation of the RMM problem is presented.

2.3.1 Fast implementation

Once the quadratic constraints have been replaced with linear constraints, the RMM is merely a quadratic program which admits many fast implementation schemes. It is now possible to adapt previous fast SVM algorithms in the literature to the RMM. In this section, the SVM^{light} [Joachims, 1998] approach will be adapted to the following RMM optimization

problem

$$\begin{aligned} \min_{\mathbf{w},b} \ & \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{n}\xi_i \\ \text{s.t. } & y_i(\mathbf{w}^\top \mathbf{x}_i + b) \geq 1 - \xi_i, \quad \xi_i \geq 0 && \forall 1 \leq i \leq n \\ & \mathbf{w}^\top \mathbf{x}_i + b \leq B && \forall 1 \leq i \leq n \\ & -\mathbf{w}^\top \mathbf{x}_i - b \leq B && \forall 1 \leq i \leq n. \end{aligned} \tag{2.8}$$

The dual of (2.8) can be shown to be the following:

$$\begin{aligned} \max_{\boldsymbol{\alpha},\boldsymbol{\lambda},\boldsymbol{\lambda}^*} \ & -\frac{1}{2}\left(\boldsymbol{\alpha} \bullet \mathbf{y} - \boldsymbol{\lambda} + \boldsymbol{\lambda}^*\right)^\top \mathbf{K}\left(\boldsymbol{\alpha} \bullet \mathbf{y} - \boldsymbol{\lambda} + \boldsymbol{\lambda}^*\right) + \boldsymbol{\alpha}^\top \mathbf{1} - B\boldsymbol{\lambda}^\top \mathbf{1} - B\boldsymbol{\lambda}^{*\top}\mathbf{1} \\ \text{s.t. } & \boldsymbol{\alpha}^\top \mathbf{y} - \boldsymbol{\lambda}^\top \mathbf{1} + \boldsymbol{\lambda}^{*\top}\mathbf{1} = 0 \\ & 0 \leq \boldsymbol{\alpha} \leq C\mathbf{1} \\ & \boldsymbol{\lambda}, \boldsymbol{\lambda}^* \geq \mathbf{0}, \end{aligned} \tag{2.9}$$

where the operator $\bullet$ denotes the element-wise product of two vectors.

The QP in (2.9) is solved in an iterative way. In each step, only a subset of the dual variables are optimized. For instance, in a particular iteration, take q, r and s ($\tilde{q}$, $\tilde{r}$ and $\tilde{s}$) to be indices of the free (fixed) variables in $\boldsymbol{\alpha}$, $\boldsymbol{\lambda}$ and $\boldsymbol{\lambda}^*$ respectively (ensuring that $q \cup \tilde{q} = \{1, 2, \ldots n\}$ and $q \cap \tilde{q} = \emptyset$ and proceeding similarly for the other two indices). The optimization over the free variables in that step can then be expressed as:

$$\begin{aligned} \max_{\boldsymbol{\alpha}_q,\boldsymbol{\lambda}_r,\boldsymbol{\lambda}_s^*} \ & -\frac{1}{2}\begin{bmatrix} \boldsymbol{\alpha}_q \bullet \mathbf{y}_q \\ \boldsymbol{\lambda}_r \\ \boldsymbol{\lambda}_s^* \end{bmatrix}^\top \begin{bmatrix} \mathbf{K}_{qq} & -\mathbf{K}_{qr} & \mathbf{K}_{qs} \\ -\mathbf{K}_{rq} & \mathbf{K}_{rr} & -\mathbf{K}_{rs} \\ \mathbf{K}_{sq} & -\mathbf{K}_{sr} & \mathbf{K}_{ss} \end{bmatrix} \begin{bmatrix} \boldsymbol{\alpha}_q \bullet \mathbf{y}_q \\ \boldsymbol{\lambda}_r \\ \boldsymbol{\lambda}_s^* \end{bmatrix} \\ & -\frac{1}{2}\begin{bmatrix} \boldsymbol{\alpha}_q \bullet \mathbf{y}_q \\ \boldsymbol{\lambda}_r \\ \boldsymbol{\lambda}_s^* \end{bmatrix}^\top \begin{bmatrix} \mathbf{K}_{q\tilde{q}} & -\mathbf{K}_{q\tilde{r}} & \mathbf{K}_{q\tilde{s}} \\ -\mathbf{K}_{r\tilde{q}} & \mathbf{K}_{r\tilde{r}} & -\mathbf{K}_{r\tilde{s}} \\ \mathbf{K}_{s\tilde{q}} & -\mathbf{K}_{s\tilde{r}} & \mathbf{K}_{s\tilde{s}} \end{bmatrix} \begin{bmatrix} \boldsymbol{\alpha}_{\tilde{q}} \bullet \mathbf{y}_{\tilde{q}} \\ \boldsymbol{\lambda}_{\tilde{r}} \\ \boldsymbol{\lambda}_{\tilde{s}}^* \end{bmatrix} \\ & + \boldsymbol{\alpha}_q^\top \mathbf{1} - B\boldsymbol{\lambda}_r^\top \mathbf{1} - B\boldsymbol{\lambda}_s^{*\top}\mathbf{1} \\ \text{s.t. } & \boldsymbol{\alpha}_q^\top \mathbf{y}_q - \boldsymbol{\lambda}_r^\top \mathbf{1} + \boldsymbol{\lambda}_s^{*\top}\mathbf{1} = -\boldsymbol{\alpha}_{\tilde{q}}^\top \mathbf{y}_{\tilde{q}} + \boldsymbol{\lambda}_{\tilde{r}}^\top \mathbf{1} - \boldsymbol{\lambda}_{\tilde{s}}^{*\top}\mathbf{1}, \\ & \mathbf{0} \leq \boldsymbol{\alpha}_q \leq C\mathbf{1}, \\ & \boldsymbol{\lambda}_r, \ \boldsymbol{\lambda}_s^* \geq \mathbf{0}. \end{aligned} \tag{2.10}$$

While the first term in the above objective is quadratic in the free variables (over which it is optimized), the second term is merely linear. Essentially, the above is a working-set scheme which iteratively solves the QP over subsets of variables until some termination criteria are achieved. The following criteria will be used to terminate the algorithm. If $\boldsymbol{\alpha}, \boldsymbol{\lambda}, \boldsymbol{\lambda}^*$ and b are the current solution (b is determined by the KKT conditions just as with SVMs), then:

$$\begin{aligned}
&\forall i \text{ s.t. } 0 < \alpha_i < C : && b - \epsilon \leq y_i - (\sum_{j=1}^{n} (\alpha_j y_j - \lambda_j + \lambda_j^*) k(\mathbf{x}_i, \mathbf{x}_j)) \leq b + \epsilon \\
&\forall i \text{ s.t. } \alpha_i = 0 : && y_i (\sum_{j=1}^{n} (\alpha_j y_j - \lambda_j + \lambda_j^*) k(\mathbf{x}_i, \mathbf{x}_j) + b) \geq 1 - \epsilon \\
&\forall i \text{ s.t. } \alpha_i = C : && y_i (\sum_{j=1}^{n} (\alpha_j y_j - \lambda_j + \lambda_j^*) k(\mathbf{x}_i, \mathbf{x}_j) + b) \leq 1 + \epsilon \\
&\forall i \text{ s.t. } \lambda_i > 0 : && B - \epsilon \leq (\sum_{j=1}^{n} (\alpha_j y_j - \lambda_j + \lambda_j^*) k(\mathbf{x}_i, \mathbf{x}_j) + b) \leq B + \epsilon \\
&\forall i \text{ s.t. } \lambda_i = 0 : && (\sum_{j=1}^{n} (\alpha_j y_j - \lambda_j + \lambda_j^*) k(\mathbf{x}_i, \mathbf{x}_j) + b) \leq B - \epsilon \\
&\forall i \text{ s.t. } \lambda_i^* > 0 : && B - \epsilon \leq -(\sum_{j=1}^{n} (\alpha_j y_j - \lambda_j + \lambda_j^*) k(\mathbf{x}_i, \mathbf{x}_j) + b) \leq B + \epsilon \\
&\forall i \text{ s.t. } \lambda_i^* = 0 : && -(\sum_{j=1}^{n} (\alpha_j y_j - \lambda_j + \lambda_j^*) k(\mathbf{x}_i, \mathbf{x}_j) + b) \leq B - \epsilon.
\end{aligned}$$

In each step of the algorithm, a small sub-problem of the structure of (2.10) is solved. To select the free variables, these conditions are checked to find the worst violating variables both from the top of the violation list and from the bottom. The selected variables are optimized by solving (2.10) while keeping the other variables fixed. Since only a small QP is solved in each step, the cubic time scaling behavior is circumvented for improved efficiency. A few other book-keeping tricks have also been adapted from SVM^{light} to yield other minor improvements.

Denote by p the number of elements chosen in each step of the optimization (i.e., $p = |q| + |r| + |s|$). The QP in each step takes $\mathcal{O}(p^3)$ and updating the prediction values to compute the KKT violations takes $\mathcal{O}(nq)$ time. Sorting the output values to choose the most violated constraints takes $\mathcal{O}(n \log(n))$ time. Thus, the total time taken in each iteration of the algorithm is $\mathcal{O}(p^3 + n \log(n) + nq)$. Empirical running times are provided in Section 2.4

for a digit classification problem.

Many other fast SVM solvers could also be adapted to the RMM. Recent advances such as the cutting plane SVM algorithm [Joachims, 2006], Pegasos [Shalev-Shwartz *et al.*, 2007] and so forth could also be adapted to solve the RMM formulation.

2.3.2 Variant of the RMM

It is not always desirable to have a parameter in a formulation that would depend explicitly on the output from a previous computation, such as B in (2.8). It is possible to overcome this issue via the following optimization problem:

$$\begin{aligned} \min_{\mathbf{w},b,\xi,t\geq 1} \ & \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{n}\xi_i + Dt & (2.11)\\ \text{s.t. } & y_i(\mathbf{w}^\top\mathbf{x}_i + b) \geq 1-\xi_i, \quad \xi_i \geq 0 & \forall 1\leq i\leq n,\\ & +(\mathbf{w}^\top\mathbf{x}_i + b) \leq t & \forall 1\leq i\leq n,\\ & -(\mathbf{w}^\top\mathbf{x}_i + b) \leq t & \forall 1\leq i\leq n. \end{aligned}$$

Note that (2.11) has a parameter D instead of the parameter B in (2.8). The two optimization problems are equivalent in the sense that for every value of B in (2.8), it is possible to have a corresponding D such that both optimization problems give the same solution.

Further, in some situations, a hard constraint bounding the outputs as in (2.11) can be detrimental due to outliers. Thus, it might be required to have a relaxation on the bounding constraints as well. This motivates the following relaxed version of (2.11):

$$\begin{aligned} \min_{\mathbf{w},b,\xi,t\geq 1} \ & \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{n}\xi_i + D(t + \frac{\nu}{n}\sum_{i=1}^{n}(\tau_i+\tau_i^*)) & (2.12)\\ \text{s.t. } & y_i(\mathbf{w}^\top\mathbf{x}_i + b) \geq 1-\xi_i, \quad \xi_i \geq 0 & \forall 1\leq i\leq n,\\ & +(\mathbf{w}^\top\mathbf{x}_i + b) \leq t+\tau_i & \forall 1\leq i\leq n,\\ & -(\mathbf{w}^\top\mathbf{x}_i + b) \leq t+\tau_i^* & \forall 1\leq i\leq n. \end{aligned}$$

In the above formulation, ν controls the fraction of outliers. It is not hard to derive the dual of the above to express it in kernelized form.

2.4 Experiments

In this section, a detailed investigation of the performance of the RMM[2] on several synthetic and real world datasets is provided.

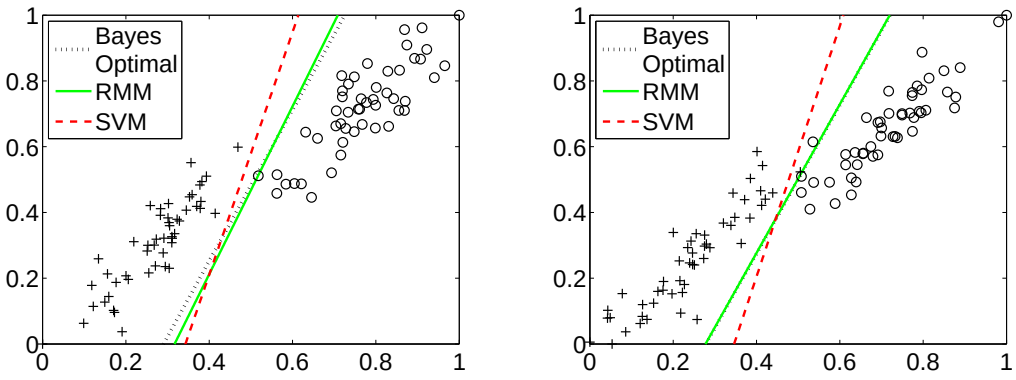

Figure 2.1: Two typical synthetic datasets (rescaled inside a 0-1 box) with corresponding SVM and RMM solutions are shown along with the Bayes optimal solution. The SVM (the RMM) solution uses the C (C and B) setting that minimized validation error. The RMM produces an estimate that is significantly closer to the Bayes optimal solution.

2.4.1 Synthetic dataset

First, consider a simple two dimensional dataset that illustrates the performance differences between the SVM and the RMM. It is possible to construct the best classifier (Bayes optimal) in this case. Consider sampling data from two different Gaussian distributions[3] corresponding to two different classes. Samples are drawn from the two following Gaussian distributions with equal prior probability:

$$\boldsymbol{\mu}_+ = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \quad \boldsymbol{\mu}_- = \begin{bmatrix} 19 \\ 13 \end{bmatrix}, \quad \boldsymbol{\Sigma} = \begin{bmatrix} 17 & 15 \\ 15 & 17 \end{bmatrix}.$$

[2]Code available at `http://www1.cs.columbia.edu/~pks2103/RMM`.

[3]Due to such Gaussian assumptions, LDA or generative modeling would be appropriate contenders but are omitted to focus the discussion on margin-based approaches.

The Gaussian distributions have different means, yet identical covariance. A total of 100,000 examples were drawn from each of Gaussian to create validation and test sets. Large validation and test sets were used to get accurate estimates of validation and test error.

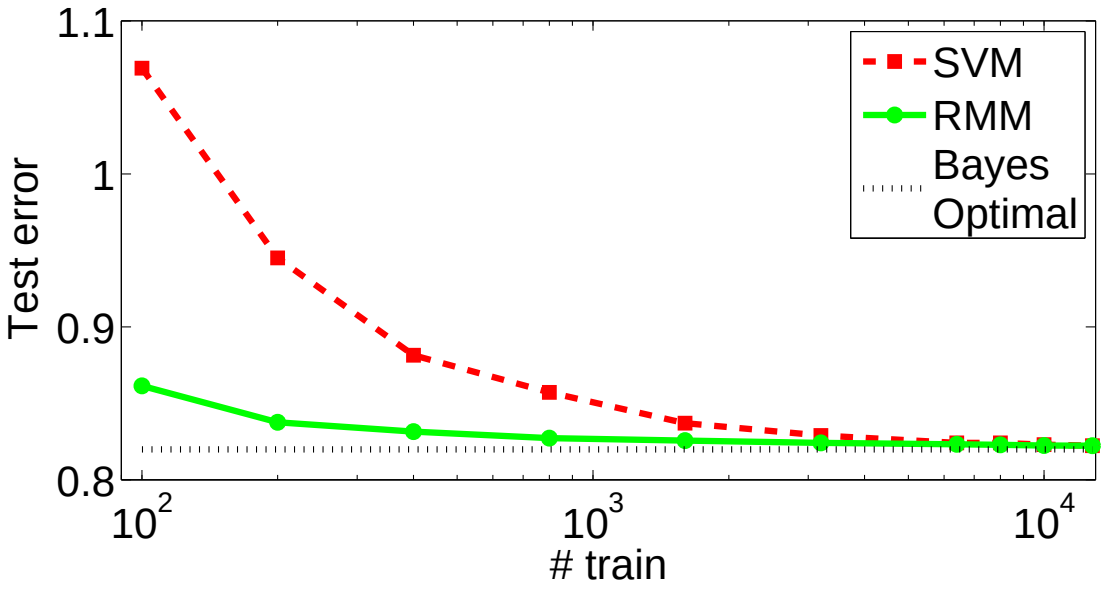

Figure 2.2: Percent test error rates for the SVM, RMM and Bayes optimal classifier as training data size is increased. The RMM has a statistically significant (at 5% level) advantage over the SVM until 6400 training examples. Subsequently, the advantage remains though with less statistical significance.

Due to the synthetic nature of the problem, the Bayes optimal classifier is easily recovered [Duda *et al.*, 2000] and is given by the following decision boundary

$$(\boldsymbol{\mu}_+ - \boldsymbol{\mu}_-)^\top \boldsymbol{\Sigma}^{-1}\mathbf{x} - 0.5(\boldsymbol{\mu}_+ - \boldsymbol{\mu}_-)^\top \boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu}_+ + \boldsymbol{\mu}_-) = 0. \tag{2.13}$$

The above formula uses the true means and covariances of the Gaussian distributions (not empirical estimates). It is clear that the Bayes optimal solution is a linear decision boundary which is in the hypothesis class explored by both the RMM and the SVM. Note that the synthetic data was subsequently normalized to lie withing the zero-one box. This rescaling was taken into account while constructing the Bayes optimal classifier (2.13).

Various C values (and B values) were explored during SVM (RMM) training. The

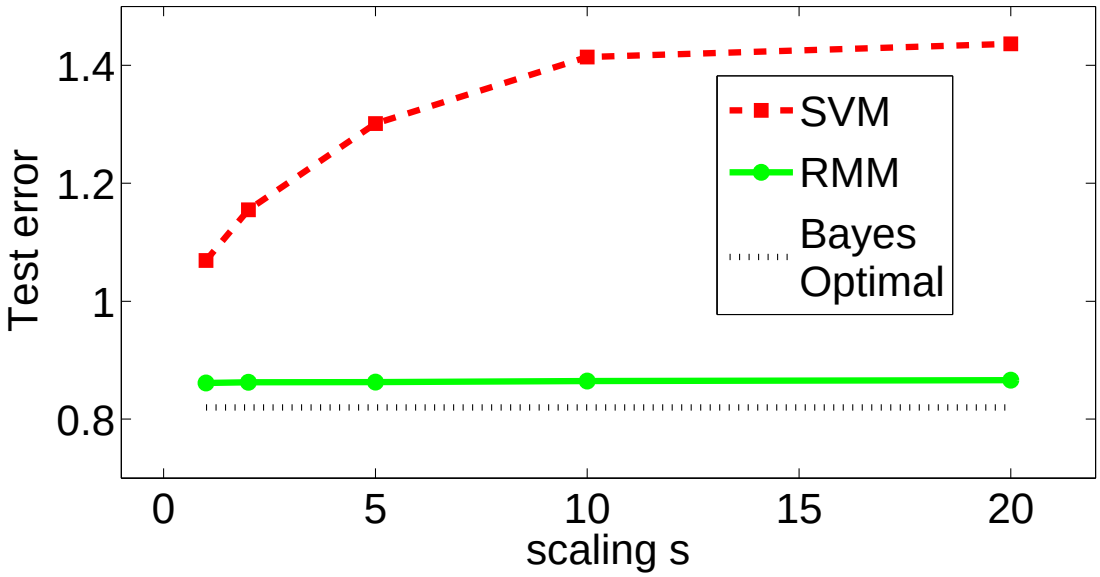

Figure 2.3: Percent test error rates for the SVM, RMM and Bayes optimal classifier as data is scaled according to (2.14). The RMM solution remains resilient to scaling while the SVM solution deteriorates significantly. The advantage of the RMM over the SVM is statistically significant (at the 1% level).

settings with minimum error rate on the validation set were used to compute test error rates. Furthermore, the test error rate for the Bayes optimal classifier was computed. Each experiment was repeated fifty times over random draws of train, test and validation sets. Figure 2.1 shows an example dataset from this synthetic experiment along with the (cross-validated) SVM, RMM and Bayes optimal classification boundaries. The SVM decision boundary is biased to separate the data in a direction where it has large spread. The RMM is less biased by the spread and is visibly closer to the Bayes optimal solution.

Figure 2.2 shows the test error rates achieved for the SVM, the RMM and the Bayes optimal classifier. The SVM performs significantly worse than the RMM, particularly when training data is scarce. The RMM maintains a statistically significant advantage over the SVM until the number of training examples grows beyond 6400. With larger training sample size n, regularization plays a smaller role in the future probability of error. This

is clear, for instance, from the bound (3.13). The last term goes to zero at $\mathcal{O}(1/\sqrt{n})$, the second term (which is the outcome of regularization) is $\mathcal{O}(\sqrt{\mathrm{tr}(\mathbf{K})/n}\sqrt{1/n})$. Both have an $\mathcal{O}(1/\sqrt{n})$ rate. However, the first term in the bound is the average slack variables divided by the margin which does not go to zero asymptotically with increasing n and eventually dominates the bound. Thus, the SVM and RMM have asymptotically similar performance but have significant differences in the small sample case.

The effect of scaling, which is a particular affine transformation, was explored next. To explore the effect of scaling in a controlled manner, first, the projection $\mathbf{w}$ recovered by the Bayes optimal classifier was obtained. An orthogonal vector $\mathbf{v}$ (such that $\mathbf{w}^\top \mathbf{v} = 0$) was then obtained. The examples (training, test and validation) were then projected onto the axes defined by $\mathbf{w}$ and $\mathbf{v}$. Each projection along $\mathbf{w}$ was preserved while the projection along $\mathbf{v}$ was scaled by a factor $s > 1$. This merely elongates the data further along directions orthogonal to $\mathbf{w}$ (*i.e.,* along the Bayes optimal classification boundary). More concisely, given an example $\mathbf{x}$, the following scaling transformation was applied:

$$\begin{bmatrix} \mathbf{w} & \mathbf{v} \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & s \end{bmatrix} \begin{bmatrix} \mathbf{w} & \mathbf{v} \end{bmatrix}^{-1} \mathbf{x}. \tag{2.14}$$

Figure 2.3 shows the SVM and RMM test error rate across a range of scaling values s. Here, 100 examples were used to construct the training data. As s grows, the SVM further deviates from the Bayes optimal classifier and attempts to separate the data along directions of large spread. Meanwhile, the RMM remains resilient to scaling and maintains a low error rate throughout.

To explore the effect of the B parameter, the average validation and test error rate were computed across many settings of C and B. The setting $C = 100$ was chosen since it obtained the minimum error rate on the validation set. The average test error rate of the RMM is shown in Figure 2.4 at $C = 100$ for multiple settings of the B parameter. Starting from the SVM solution on the right (i.e. large B) the error rate begins to fall until it attains a minimum and then starts to go increase. A similar behavior is seen in many real world datasets. Surprisingly, some datasets even exhibit monotonic reduction in test error as the value of B is decreased. The following section investigates such real world experiments in more detail.

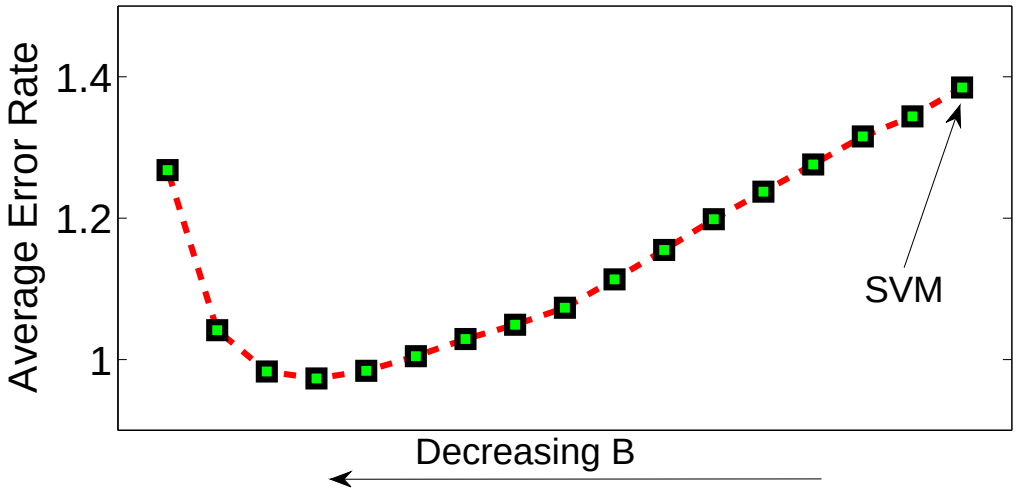

Figure 2.4: Behavior on the toy dataset with $C = 100$. As the B value is decreased, the error rate decreases to a reasonably wide minimum before starting to increase.

2.4.2 Experiments on digits

Experiments were carried out on three datasets of digits - optical digits from the UCI machine learning repository [Asuncion and Newman, 2007], USPS digits [LeCun *et al.*, 1989] and MNIST digits [LeCun *et al.*, 1998]. These datasets vary considerably in terms of their number of features (64 in optical digits, 256 in USPS and 784 in MNIST) and their number of training examples (3823 in optical digits, 7291 in USPS and 60000 in MNIST). In all the multi-class experiments, the one versus one classification strategy was used. The one versus one strategy trains a classifier for every combination of two classes. The final prediction for an example is simply the class that is predicted most often. These results are directly comparable with various methods that have been applied on this dataset. For a fair comparison, results from contender methods that use special preprocessing or domain knowledge are not explored in this thesis.[4]

In all experiments, the digits were first normalized to have unit norm. This eliminates numerical problems that may arise in kernel functions such as the polynomial kernel $k(\mathbf{u}, \mathbf{v}) = (1 + \mathbf{u}^\top \mathbf{v})^d$. Classification results were then examined for various degrees of the

[4]Additional results are reported in `http://yann.lecun.com/exdb/mnist/`.

		1	2	3	4	5	6	7	RBF
OPT	SVM	71	57	54	47	40	46	46	51
	Σ-SVM	**61**	48	41	36	35	31	**29**	47
	KLDA	71	57	54	47	40	46	46	**45**
	RMM	71	**36**	**32**	**31**	**33**	**30**	**29**	51
USPS	SVM	145	109	109	103	100	95	93	104
	Σ-SVM	**132**	**108**	99	94	**89**	**87**	**90**	**97**
	KLDA	132	119	121	117	114	118	117	101
	RMM	153	109	**94**	**91**	91	90	**90**	98
1000-MNIST	SVM	696	511	422	380	362	338	332	670
	Σ-SVM	**671**	470	373	341	322	309	303	673
	KLDA	1663	848	591	481	430	419	405	1597
	RMM	689	**342**	**319**	**301**	**298**	**290**	**296**	**613**
2/3-MNIST	SVM	552	237	200	183	178	177	164	166
	RMM	**534**	**164**	**148**	**140**	**123**	**129**	**129**	**144**
Full MNIST	SVM	536	198	170	156	157	141	136	146
	RMM	**521**	**146**	**140**	**130**	**119**	**116**	**115**	**129**

Table 2.1: The number of misclassification in three different digit datasets. Various kernels are explored using the SVM, **Σ**-SVM, KLDA and RMM methods.

polynomial kernel. In addition, kernel values were further normalized so that the trace of the training Gram matrix was equal to the number of training examples.

All parameters were tuned by splitting the training data according to an 80:20 ratio with the larger split being used for training and the smaller split for validation. The process was repeated five times over random splits to select hyper-parameters (C for the SVM, C and D for the **Σ**-SVM and C and B for the RMM). A final classifier was trained for each of the 45 classification problems with the best parameters found by cross validation using all the training examples in its corresponding pair of classes.

For the MNIST digits experiment, the **Σ**-SVM and kernel LDA (KLDA) methods were too computationally demanding due to their use of matrix inversion. To cater to these meth-

ods, a smaller experiment was conducted with 1000 examples per training. For the larger experiments, the $\mathbf{\Sigma}$-SVM and KLDA were excluded. The larger experiment on MNIST involved training on two thirds of the digits (i.e. training with an average of 8000 examples for each pair of digits) for each binary classification task. In both these experiments, the remaining training data was used as a validation set. The classifier that performed best on the validation set was used for testing.

After forming all 45 classifiers (corresponding to each pair of digits), testing was done on the standard separate test sets available for each of these three benchmark problems (1797 examples in the case of optical digits, 2007 examples in USPS and 10000 examples in MNIST). The final prediction for each test example was recovered based on the majority of predictions made by the 45 classifiers on the test example with ties broken uniformly at random.

It is important to note that, on the MNIST test set, an error rate improvement of 0.1% has been established as statistically significant [Bengio *et al.*, 2007; Decoste and Schölkopf, 2002]. This corresponds to 10 or more test examples being correctly classified by one method over an other.

Table 2.1 shows results on all three digits datasets for polynomial kernels under varying degrees as well as for RBF kernels. For each dataset, the number of misclassified examples using the majority voting scheme above is reported. The $\mathbf{\Sigma}$-SVM typically outperforms the SVM yet the RMM outperforms both. Interestingly, with higher degree kernels, the $\mathbf{\Sigma}$-SVM seems to match the performance of the RMM while in most lower-degree kernels, the RMM outperforms both the SVM and the $\mathbf{\Sigma}$-SVM convincingly. Since the $\mathbf{\Sigma}$-SVM is prohibitive to run on large scale datasets due to the computationally cumbersome matrix inversion, the RMM was clearly the most competitive method in these experiments in terms of both accuracy and computational efficiency.

The best parameters found by validation in the previous experiments were used in a full-scale MNIST experiment which does not have a validation set of its own. All 45 pairwise classifiers (both SVMs and RMMs) were trained with the previously cross-validated parameters using *all* the training examples for each class in MNIST for various kernels. The test results are reported in Table 2.1; the RMM advantages persist in this full-scale MNIST

experiment.

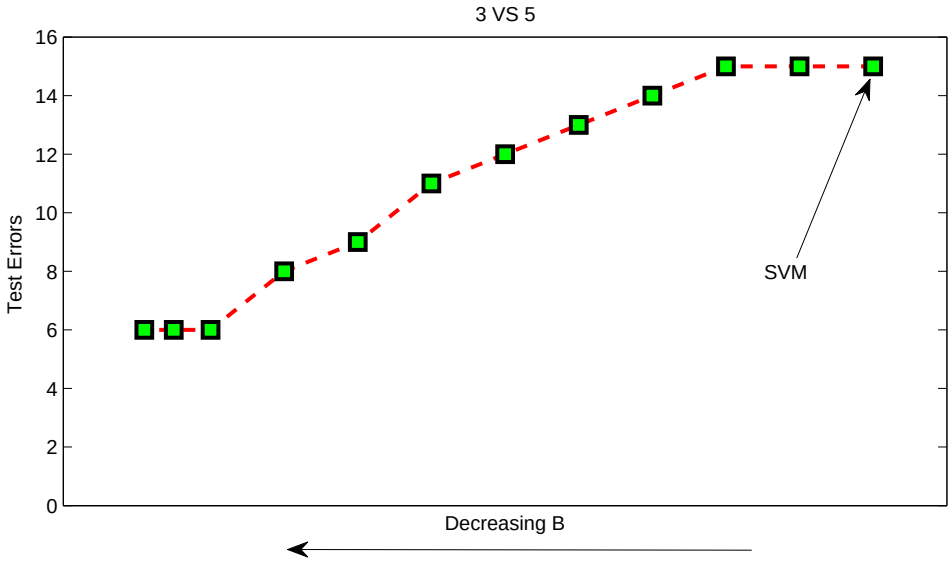

Figure 2.5: Performance on MNIST test set with digits 3 and 5. The number of errors decreases from 15 to 6 as B decreases from the right.

2.4.3 Classifying MNIST digits 3 vs 5

This section presents more detailed results on one particular binary classification problem in the MNIST digits dataset: the classification of digit 3 versus 5. Therein, the RMM has a dramatically stronger performance than the SVM. The results reported in this section are with polynomial kernels of degree 5. The parameter C was selected as mentioned above. With the selected C value, an SVM was first trained over the entire MNIST training set containing the digits 3 and 5. After noting the maximum absolute value of the output given on all the training examples, B value was reduced in steps. The number of test errors on the MNIST test set (3 versus 5) was noted. As the B value is reduced, the number of errors starts to diminish as shown in Figure 2.5. The number of errors produced by the SVM was 15. With the RMM, the number of errors dropped to 6 as the B value approached one. Clearly, as B decreases, the absolute margin is decreased however the test error rate drops drastically. This empirically suggests that maximizing the relative margin can have

a beneficial effect on the performance of a classifier. Admittedly, this is only one example and is provided only for illustrative purposes. However, similar behavior was observed in most of the binary digit classification problems though in some cases the error rate did not go down significantly with decreasing B values. The generalization behavior on all 45 individual problems is explored in more detail in Section 2.4.4.

2.4.4 All 45 binary MNIST problems

This section explores RMM performance on the 45 pairwise digit classification problems in isolation. In these experiments, both C and B values were fixed using validation as in previous sections. A total of 45 binary classifiers were constructed using all MNIST training digits. The resulting error rates are shown in Figure 2.6. On most problems, the RMM obtains a significantly lower error rate than the SVM and, at times obtains half the error rate.

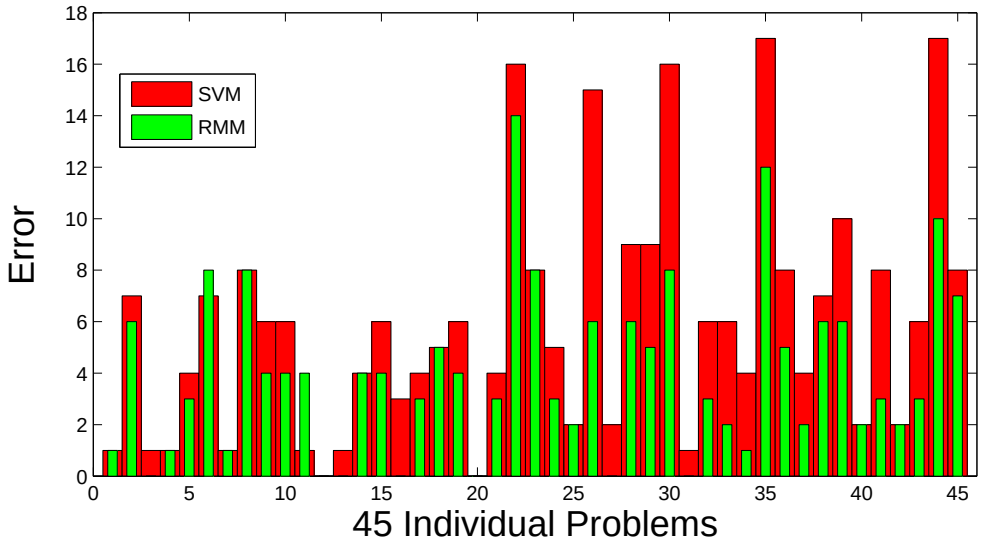

Figure 2.6: Total test errors on all 45 MNIST classification problems. Various classifiers were trained on the entire MNIST training dataset and evaluated on a standardized separate test set.

Method	RMM	$\mathfrak{U}$-SVM		
# Universum	-	1000	3000	all
Error rate	1.081	1.059	1.037	1.020
Error Std Dev	0.138	0.142	0.149	0.159
p-value		0.402	0.148	0.031

Table 2.2: Percentage error rates for the RMM and the $\mathfrak{U}$-SVM. The rate for the SVM was 1.274 with a standard deviation of 0.179; this is significantly larger than all other results in the table (with a p-value of 0.000). The final row reports the p-value of a paired t-test between the RMM error rate and the $\mathfrak{U}$-SVM error rate (corresponding to the Universum size being considered in that column).

2.4.4.1 A comparison with the Universum method

A new framework known as the Universum [Weston *et al.*, 2006; Sinz *et al.*, 2008] was recently introduced which maximizes the margin while keeping classifier outputs low on an additional collection of non-examples that do not belong to either class of interest. These additional examples are known as Universum examples; these examples are obtained from any other distribution other than the one generating the training data. In the RMM, classification outputs on training examples are bounded; in the Universum, classification outputs on Universum examples are bounded (albeit with a different loss). The following experiments compare the Universum based framework with the RMM.

An MNIST experiment was explored for classifying digits 5 vs 8 using 1000 labeled training examples under the RBF kernel. This setup is identical to the experimental conditions described in [Weston *et al.*, 2006]. Examples of the digit 3 served as Universum examples since these were reported to be the *best* performing Universum examples in previous work [Weston *et al.*, 2006]. The experiments used the standard implementation of the Universum provided by the authors [Weston *et al.*, 2006] under the default parameter settings (for variables such as ϵ). The Universum was compared with the RMM which had access to the same 1000 training examples. Furthermore, 3500 examples were used as a test set and another 3500 examples as a validation set to perform model selection. All parameter settings for the

RMM and the Universum SVM (or $\mathfrak{U}$-SVM) as well as the variance parameter of the RBF kernel were explored over a wide range of values. The parameter settings that achieved the smallest error on a validation set were then used to evaluate performance on the test set (and vice-versa). This entire experiment was repeated ten-fold over different random draws of the various sets. The average test error rates were compiled for both algorithms.

While the RMM only had access to the 1000 training examples, the $\mathfrak{U}$-SVM was also given a Universum of images of the digit 3. The Universum spanned three different sizes - 1000, 3000 and 6131 examples (i.e. all available images of the digit 3 in the MNIST training set). The results are reported in Table 2.2. First, observe that both the RMM and the $\mathfrak{U}$-SVM improved the baseline SVM performance significantly (as measured by a paired t-test). With 1000 and 3000 Universum examples, even though the error rate of the $\mathfrak{U}$-SVM was slightly lower, a paired t-test revealed that it did not achieve statistically significant improvement over the RMM. Statistically significant advantages for the $\mathfrak{U}$-SVM only emerged when *all* the available images of the digit 3 were used in the Universum.

Note that there is a slight discrepancy between the errors reported here and those in [Weston *et al.*, 2006] even though both methods used the digit 3 to generate Universum examples. This may be because the previous authors [Weston *et al.*, 2006] reported the *best test error* on 1865 examples. In this thesis, a more conservative approach is taken where a good model is first selected using the validation set and then errors are reported on an unseen test set without further tuning. Clearly, picking the minimum error rate on a test set will give more optimistic results but tuning to the test set can be potentially misleading. This makes it difficult to directly compare test error rates with those reported in the previous paper. While the error rate (using all digits 3 as the Universum examples) in our experiments varied from 0.74% to 1.35%, the authors in [Weston *et al.*, 2006] reported an error rate of 0.62%.

With 1000 training examples, the RMM (as in Equation (2.6)) has 1000 classification constraints and 1000 bounding constraints. With 1000 Universum examples, the $\mathfrak{U}$-SVM also has 1000 bounding constraints in addition to the classification constraints. It is interesting to note that the RMM, with no extra data, is not significantly worse than a $\mathfrak{U}$-SVM endowed with an additional 1000 or 3000 *best-possible* Universum examples.

The authors of [Weston *et al.*, 2006] observed that Universum examples help most when they are correlated with the training examples. This, coupled with the results in Table 2.2 and the fact that training examples are correlated most with themselves (or with examples from the same distribution as the training examples), raises the following question: How much of the performance gain with the $\mathfrak{U}$-SVM is due to the extra examples and how much of it is due to its implicit control of the spread (as with an RMM)? This is left as an open question in this thesis and as motivation for further theoretical work.

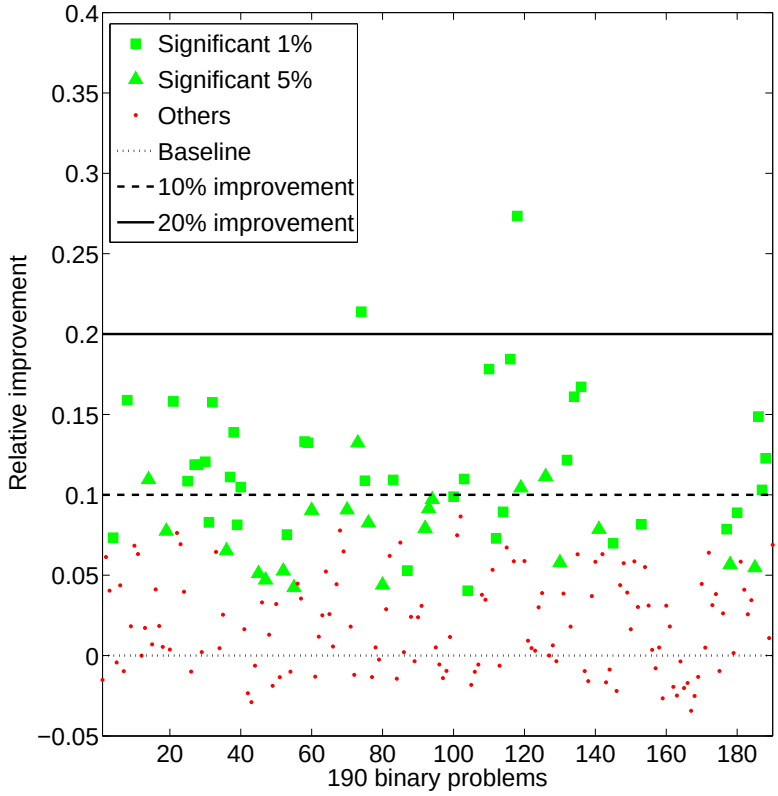

Figure 2.7: Percentage improvement of the RMM over the SVM on all 190 binary problems. Significance tests were performed using a paired t-test at the indicated levels of significance. On most problems, the RMM shows significant improvement over SVM.

2.4.5 Text classification

In this section, results are reported on the 20 Newsgroups [5] dataset. This dataset has posts from 20 different Usenet newsgroups. Each post was represented by a vector which counts the number of words that occurred in the document. In the text classification literature, this is commonly known as the bag of words representation. Each feature vector was divided by the total number of words in the document to normalize it.

All 190 binary pairwise classification problems were considered in this experiment. For each problem, 500 examples were used for training. The remaining examples were divided into a validation and test set of the same size. Both SVMs and RMMs were trained for various values of their parameters. After finding the parameter settings that achieved the lowest error on a validation set, the test error was evaluated (and vice-versa). This experiment was repeated ten times for random draws of the train, validation and test sets.

Figure 2.7 summarizes the results. For each binary classification problem, a paired t-test was performed and p-values were obtained. As can be seen from the plot, the RMM outperforms the SVM significantly in almost 30% of the problems. This experiment once again demonstrates that an absolute margin does not always result in a small test error.

2.4.6 Benchmark datasets

To compare the performance of the RMM with a number of other methods, experiments were performed on several benchmark datasets. In particular, 100 training and test splits of 13 of these datasets have been previously used in [Raetsch *et al.*, 2001; Mika *et al.*, 1999; Cawley and Talbot, 2003].[6] The RBF kernel was used in these experiments for all kernel-based methods. To handle the noisy nature of these datasets, the kernelized and relaxed version of the RMM (2.12) was used. All the parameters were tuned using cross-validation using a similar setup as in [Raetsch *et al.*, 2001][7]. With the chosen values of these parameters, the error rates were first obtained for all 100 test splits using the corresponding

[5] This dataset is available online at `http://people.csail.mit.edu/jrennie/20Newsgroups/`.

[6] These datasets are available at `http://theoval.cmp.uea.ac.uk/~gcc/matlab/default.html#benchmarks`.

[7] The values of the selected parameters and the code for the RMM are available for download at `http://www.cs.columbia.edu/~pks2103/ucirmm/`.

Dataset	SVM	KFDA	Σ-SVM	RMM (C=D)	RMM
banana	10.5 ± 0.4	10.8 ± 0.5	10.5 ± 0.4	**10.4 ± 0.4**	**10.4 ± 0.4***
b.cancer	**25.3 ± 4.6***	26.6 ± 4.8	28.8 ± 4.6	25.9 ± 4.5	**25.4 ± 4.6**
diabetes	**23.1 ± 1.7**	**23.2 ± 1.8**	24.2 ± 1.9	**23.1 ± 1.7**	**23.0 ± 1.7***
f.solar	**32.3 ± 1.8**	33.1 ± 1.6	34.6 ± 2.0	**32.3 ± 1.8***	**32.3 ± 1.8***
German	**23.4 ± 2.2**	24.1 ± 2.4	25.9 ± 2.4	**23.4 ± 2.1**	**23.2 ± 2.2***
heart	15.5 ± 3.3	15.7 ± 3.2	19.9 ± 3.6	15.4 ± 3.3	**15.2 ± 3.1***
image	**3.0 ± 0.6**	3.1 ± 0.6	3.3 ± 0.7	3.0 ± 0.6	**2.9 ± 0.7**
ringnorm	1.5 ± 0.1	**1.5 ± 0.1**	1.5 ± 0.1	**1.5 ± 0.1**	**1.5 ± 0.1***
splice	10.9 ± 0.7	10.6 ± 0.7	10.8 ± 0.6	10.8 ± 0.6	10.8 ± 0.6
thyroid	4.7 ± 2.1	**4.2 ± 2.1**	4.5 ± 2.1	**4.2 ± 1.8***	**4.2 ± 2.2**
titanic	22.3 ± 1.1	**22.0 ± 1.3***	23.1 ± 2.2	22.3 ± 1.1	22.3 ± 1.0
twonorm	**2.4 ± 0.1***	**2.4 ± 0.2**	2.5 ± 0.2	2.4 ± 0.1	**2.4 ± 0.1**
waveform	9.9 ± 0.4	9.9 ± 0.4	10.5 ± 0.5	10.0 ± 0.4	**9.7 ± 0.4***

Dataset	RBF	AB	LPAB	QPAB	ABR
banana	10.8 ± 0.4	12.3 ± 0.7	10.7 ± 0.4	10.9 ± 0.5	10.9 ± 0.4
b.cancer	27.6 ± 4.7	30.4 ± 4.7	26.8 ± 6.1	**25.9 ± 4.6**	26.5 ± 4.5
diabetes	24.3 ± 1.9	26.5 ± 2.3	24.1 ± 1.9	25.4 ± 2.2	23.8 ± 1.8
f.solar	34.4 ± 1.9	35.7 ± 1.8	34.7 ± 2.0	36.2 ± 1.8	34.2 ± 2.2
German	24.7 ± 2.4	27.5 ± 2.5	24.8 ± 2.2	25.3 ± 2.1	24.3 ± 2.1
heart	17.1 ± 3.3	20.3 ± 3.4	17.5 ± 3.5	17.2 ± 3.4	16.5 ± 3.5
image	3.3 ± 0.7	**2.7 ± 0.7**	2.8 ± 0.6	**2.7 ± 0.6***	**2.7 ± 0.6***
ringnorm	1.7 ± 0.2	1.9 ± 0.2	2.2 ± 0.5	1.9 ± 0.2	1.6 ± 0.1
splice	9.9 ± 0.8	10.1 ± 0.5	10.2 ± 1.6	10.1 ± 0.5	**9.5 ± 0.6***
thyroid	**4.5 ± 2.1**	**4.4 ± 2.2**	**4.6 ± 2.2**	**4.3 ± 2.2**	4.5 ± 2.2
titanic	23.3 ± 1.3	22.6 ± 1.2	24.0 ± 4.4	22.7 ± 1.0	22.6 ± 1.2
twonorm	2.8 ± 0.3	3.0 ± 0.3	3.2 ± 0.4	3.0 ± 0.3	2.7 ± 0.2
waveform	10.7 ± 1.1	10.8 ± 0.6	10.5 ± 1.0	10.1 ± 0.5	**9.8 ± 0.8**

Table 2.3: UCI results for a number of classification methods. Results are shown for the SVM, regularized kernel Fisher Discriminant Analysis, the Σ-SVM, the RMM, an RBF network, Adaboost, LP-regularized Adaboost, QP-regularized Adaboost and Regularized Adaboost. The results have been split into two parts due to lack of space. For each dataset, the algorithm which gave the minimum error rate is starred. All other algorithms that were not significantly different from (at the 5% significance level based on a paired t-test) the minimum error rate are in boldface.

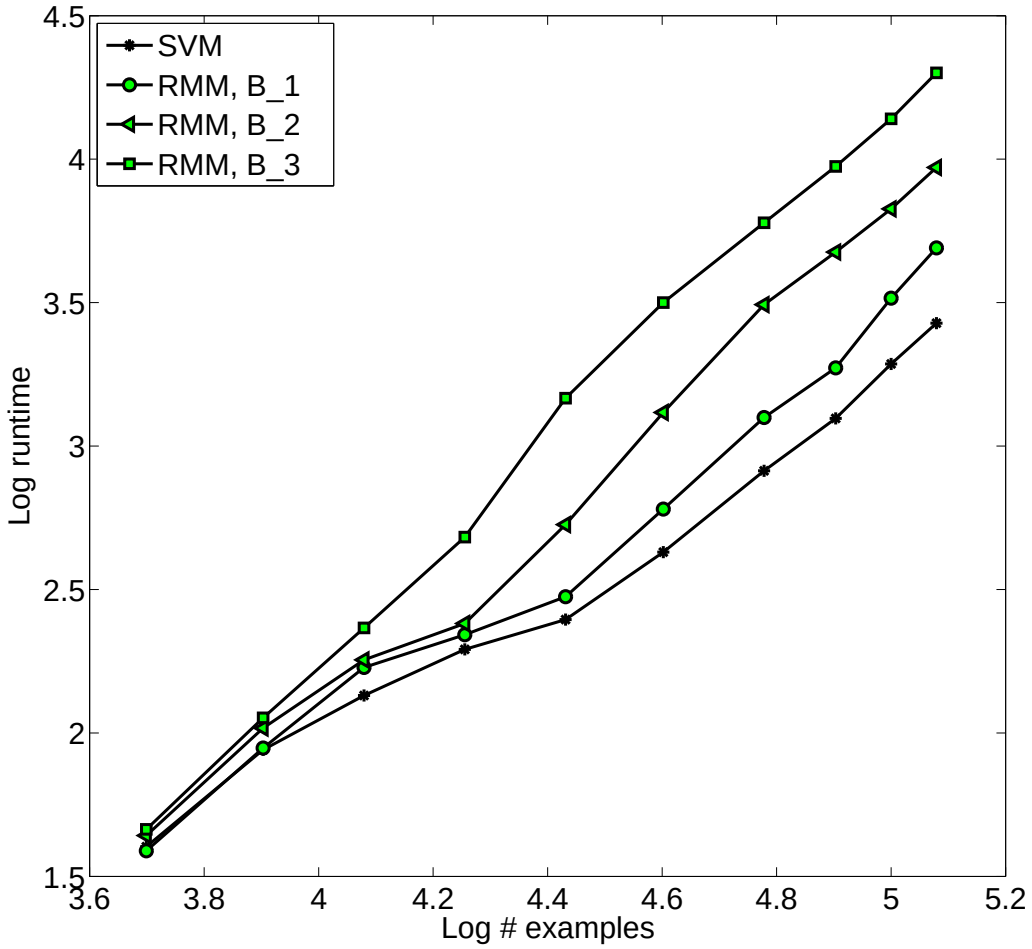

Figure 2.8: Log run time versus log number of examples. The figure shows that the SVM and the RMM have similar computational requirements overall.

training splits. The results are reported in Table 2.3. Once again, the RMM exhibits clear performance advantages over other methods.

2.4.7 Scalability and run-time

While the asymptotic run time behavior was analyzed in Section 2.3.1, the run time of the RMM is also studied empirically in this section. In particular, the classification of MNIST

digits 0-4 versus 5-9 with a polynomial kernel of degree five was used to benchmark the algorithms. For both the RMM and the SVM, the tolerance parameter (ϵ mentioned in Section 2.3.1) was set to 0.001. The size of the sub-problem (2.10) solved was 800 in all the cases. To evaluate how the algorithms scale, the number of training examples was increased in steps and the training time was noted. Throughout all the experiments, the C value was set to 1. The SVM was first run on the training examples. The value of maximum absolute prediction θ was noted. Three different values of B were then tried for the RMM: $B_1 = 1 + (\theta - 1)/2$, $B_2 = 1 + (\theta - 1)/4$ and $B_3 = 1 + (\theta - 1)/10$. In all experiments, the run time was noted. The experiment was repeated ten times to get an average run time for each B value. A log-log plot comparing the number of examples to the average run time is shown in Figure 2.8. Both the SVM and the RMM run time exhibit similar asymptotic behavior.

2.5 Summary

Several formulations were discussed in this chapter to overcome the sensitives of the support vector machines discussed in Chapter 1. While there are several ways to overcome the sensitivities, the RMM is particularly suited due to computational and generalization reasons (topic of the next chapter). Extensive experimentation on synthetic as well as real world datasets showed that RMM can significantly outperform the SVM.

Chapter 3

Risk Bounds

This chapter provides generalization guarantees for the classifiers of interest (the SVM, $\mathbf{\Sigma}$-SVM and RMM) which all produce decision[1] boundaries of the form $\mathbf{w}^\top \mathbf{x} = 0$ from a limited number of examples. In the case of SVM, the decision boundary is found by minimizing a combination of $\mathbf{w}^\top \mathbf{w}$ and an upper bound on the number of errors. This minimization is equivalent to choosing a function $g(\mathbf{x}) = \mathbf{w}^\top \mathbf{x}$ from a set of linear functions with bounded ℓ_2 norm. Therefore, with a suitable choice of E, the SVM solution chooses the function $g(\cdot)$ from the set $\{\mathbf{x} \to \mathbf{w}^\top \mathbf{x} | \frac{1}{2}\mathbf{w}^\top \mathbf{w} \leq E\}$.

By measuring the complexity of the function class being explored, it is possible to derive generalization guarantees. A natural measure of how complex a function class is the Rademacher complexity which has been fruitful in the derivation of generalization bounds. For SVMs, such results can be found in [Bartlett and Mendelson, 2002; Shawe-Taylor and Cristianini, 2004]. This chapter continues in the same spirit and defines the function classes and their corresponding Rademacher complexities for slightly modified versions of the RMM as well as the $\mathbf{\Sigma}$-SVM. Furthermore, these will be used to provide generalization guarantees for both classifiers. The style and content of this section closely follows that of [Shawe-Taylor and Cristianini, 2004]. These risk bounds were first published in [Shivaswamy and Jebara, 2010b].

The function classes for the RMM and $\mathbf{\Sigma}$-SVM will depend on the data. Thus, these

[1]The bias term is suppressed in this section for brevity.

both entail so-called data-dependent regularization which is not quite as straightforward as the function classes explored by SVMs. In particular, the data involved in defining data-dependent function classes will be treated differently and referred to as landmarks to distinguish them from the training data. Landmark data is used to define the function class while training data is used to select a specific function from the class. This distinction is important for the following theoretical derivations. However, in practical implementations, both the $\mathbf{\Sigma}$-SVM and the RMM may use the training data to both define the function class and to choose the best function within it. Thus, the distinction between landmark data and training data is merely a formality for deriving generalization bounds which require independent sets of examples for both stages. Ultimately, however, it will be possible to still provide generalization guarantees that are independent of the particular landmark examples. Details of this argument are provided in Section 3.3. At the outset, however, it is assumed that, in addition to the training data, a separate dataset of landmarks is provided to define the function class for the RMM and the $\mathbf{\Sigma}$-SVM.

3.1 Function class definitions

Consider the training data set $(\mathbf{x}_i, y_i)_{i=1}^n$ with $\mathbf{x}_i \in \mathbb{R}^m$ and $y_i \in \{\pm 1\}$ which are drawn independently and identically distributed (*iid*) from an unknown underlying distribution $\mathbf{P}[(\mathbf{x}, y)]$ denoted as $\mathcal{D}$. The features of the training examples above are denoted by the set $\mathbf{S} = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$.

Given a choice of the parameter E in the SVM (where E plays the role of the regularization parameter), the set of linear functions the SVM considers is:

Definition 3 $\mathcal{F}_E := \{\mathbf{x} \to \mathbf{w}^\top \mathbf{x} | \frac{1}{2}\mathbf{w}^\top \mathbf{w} \leq E\}$.

The RMM maximizes the margin while also limiting the spread of projections on the training data. It effectively considers the following function class:

Definition 4 $\mathcal{H}_{E,D}^S := \{\mathbf{x} \to \mathbf{w}^\top \mathbf{x} | \frac{\bar{D}}{2}\mathbf{w}^\top \mathbf{w} + \frac{D}{2}(\mathbf{w}^\top \mathbf{x}_i)^2 \leq E \ \forall 1 \leq i \leq n\}$.

In the above equation, $\bar{D} := 1 - D$ and $0 < D < 1$ trade off between large margin and

small spread on the projections.[2] Since the above function class depends on the training examples, standard Rademacher analysis, which is straightforward for the SVM, is no longer applicable. Instead, define another function class for the RMM using a distinct set of landmark examples.

A set $\mathbf{V} = \{\mathbf{v}_1, \ldots, \mathbf{v}_{n_v}\}$ drawn *iid* from the same distribution $\mathbf{P}[\mathbf{x}]$, denoted as $\mathcal{D}_x$, is used as the landmark examples. With these landmark examples, the modified RMM function class can be written as:

Definition 5 $\mathcal{H}^V_{E,D} := \{\mathbf{x} \to \mathbf{w}^\top \mathbf{x} | \frac{\bar{D}}{2}\mathbf{w}^\top\mathbf{w} + \frac{D}{2}(\mathbf{w}^\top \mathbf{v}_i)^2 \leq E \ \forall 1 \leq i \leq n_v\}$.

Finally, function classes that are relevant for the $\mathbf{\Sigma}$-SVM are considered. These limit the average projection rather than the maximum projection. The data-dependent function class is defined as below:

Definition 6 $\mathcal{G}^S_{E,D} := \{\mathbf{x} \to \mathbf{w}^\top \mathbf{x} | \frac{\bar{D}}{2}\mathbf{w}^\top\mathbf{w} + \frac{D}{2n}\sum_{i=1}^n (\mathbf{w}^\top \mathbf{x}_i)^2 \leq E\}$.

A different landmark set $\mathbf{U} = \{\mathbf{u}_1, \ldots, \mathbf{u}_n\}$, again drawn *iid* from $\mathcal{D}_x$, is used in defining the corresponding landmark function class:

Definition 7 $\mathcal{G}^U_{B,D} := \{\mathbf{x} \to \mathbf{w}^\top \mathbf{x} | \frac{\bar{D}}{2}\mathbf{w}^\top\mathbf{w} + \frac{D}{2n}\sum_{i=1}^n (\mathbf{w}^\top \mathbf{u}_i)^2 \leq B\}$.

Note that the parameter E is fixed in $\mathcal{H}^V_{E,D}$ but n_v may be different from n. In the case of $\mathcal{G}^U_{B,D}$, the number of landmarks is the same (n) as the number of training examples but the parameter B is used instead of E. These distinctions are intentional and will be clarified in subsequent sections.

3.2 Rademacher complexity

In this section, the Rademacher complexity of the aforementioned function classes are quantified by bounding the empirical Rademacher complexity. Rademacher complexity measures the richness of a class of real-valued functions with respect to a probability distribution [Bartlett and Mendelson, 2002; Shawe-Taylor and Cristianini, 2004; Bousquet *et al.*, 2004].

[2] Zero and one are excluded from the range of D to avoid degenerate cases.

Definition 8 *For a sample* $\mathbf{S} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$ *generated by a distribution on* $\mathbf{x}$ *and a real-valued function class* $\mathcal{F}$ *with domain* $\mathbf{x}$*, the empirical Rademacher complexity*[3] *of* $\mathcal{F}$ *is*

$$\hat{R}(\mathcal{F}) := \mathbf{E}_\sigma \left[\sup_{f \in \mathcal{F}} \left| \frac{2}{n} \sum_{i=1}^{n} \sigma_i f(\mathbf{x}_i) \right| \right]$$

where $\sigma = \{\sigma_1, \ldots \sigma_n\}$ *are independent random variables that take values* $+1$ *or* -1 *with equal probability. Moreover, the Rademacher complexity of* $\mathcal{F}$ *is:* $R(\mathcal{F}) := \mathbf{E}_\mathbf{S} \left[\hat{R}(\mathcal{F}) \right]$.

A stepping stone for quantifying the true Rademacher complexity is obtained by considering its empirical counterpart.

3.2.1 Empirical Rademacher complexity

In this section, upper bounds on the empirical Rademacher complexities are derived for the previously defined function classes. These bounds provide insights on the regularization properties of the function classes for the sample $\mathbf{S} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots \mathbf{x}_n\}$.

Theorem 9 $\hat{R}(\mathcal{F}_E) \leq T_0 := \frac{2\sqrt{2E}}{n} \sqrt{tr(\mathbf{K})}$, *where* $tr(\mathbf{K})$ *is the trace of the Gram matrix of the elements in* $\mathbf{S}$.

Proof

$$\begin{aligned}
\hat{R}(\mathcal{F}_E) &= \mathbf{E}_\sigma \left[\sup_{f \in \mathcal{F}_E} \left| \frac{2}{n} \sum_{i=1}^{n} \sigma_i f(\mathbf{x}_i) \right| \right] = \frac{2}{n} \mathbf{E}_\sigma \left[\max_{||\mathbf{w}|| \leq \sqrt{2E}} \left| \mathbf{w}^\top \sum_{i=1}^{n} \sigma_i \mathbf{x}_i \right| \right] \\
&\leq \frac{2\sqrt{2E}}{n} \mathbf{E}_\sigma \left[\left| \left| \sum_{i=1}^{n} \sigma_i \mathbf{x}_i \right| \right| \right] = \frac{2\sqrt{2E}}{n} \mathbf{E}_\sigma \left[\left(\sum_{i=1}^{n} \sigma_i \mathbf{x}_i^\top \sum_{j=1}^{n} \sigma_j \mathbf{x}_j \right)^{\frac{1}{2}} \right] \\
&\leq \frac{2\sqrt{2E}}{n} \left(\mathbf{E}_\sigma \left[\sum_{i,j=1}^{n} \sigma_i \sigma_j \mathbf{x}_i^\top \mathbf{x}_j \right] \right)^{\frac{1}{2}} = \frac{2\sqrt{2E}}{n} \sqrt{\mathrm{tr}(\mathbf{K})}.
\end{aligned}$$

The proof uses Jensen's inequality on the function $\sqrt{\cdot}$ and the fact that σ_i and σ_j are random variables taking values $+1$ or -1 with equal probability. Thus, when $i \neq j$, $\mathbf{E}_\sigma[\sigma_i \sigma_j \mathbf{x}_i^\top \mathbf{x}_i] = 0$ and, otherwise, $\mathbf{E}_\sigma[\sigma_i \sigma_i \mathbf{x}_i^\top \mathbf{x}_i] = \mathbf{E}_\sigma[\mathbf{x}_i^\top \mathbf{x}_i] = \mathbf{x}_i^\top \mathbf{x}_i$. The result follows

[3] The dependence of the empirical Rademacher complexity on n and $\mathbf{S}$ is suppressed by writing $\hat{R}(\mathcal{F})$ for brevity.

from the linearity of the expectation operator. ■

Roughly speaking, by keeping E small, the classifier's ability to fit arbitrary labels is reduced. This is one way to motivate a maximum margin strategy. Note that $\sqrt{\mathrm{tr}(\mathbf{K})}$ is a coarse measure of the spread of the data. However, most SVM formulations do not directly optimize this term. This motivates to next consider two new function classes.

Theorem 10 *$\hat{R}(\mathcal{H}^V_{E,D}) \leq T_2(\mathbf{V},\mathbf{S})$, where for any training set $\mathcal{B}$ and landmark[4] set $\mathcal{A}$, $T_2(\mathcal{A},\mathcal{B}) := \min_{\lambda\geq 0}\ \frac{1}{|\mathcal{B}|}\sum_{\mathbf{x}\in\mathcal{B}}\mathbf{x}^\top\left(\bar{D}\mathbf{I}\sum_{\mathbf{u}\in\mathcal{A}}\lambda_{\mathbf{u}} + D\sum_{\mathbf{u}\in\mathcal{A}}\lambda_{\mathbf{u}}\mathbf{u}\mathbf{u}^\top\right)^{-1}\mathbf{x} + \frac{2E}{|\mathcal{B}|}\sum_{\mathbf{u}\in\mathcal{A}}\lambda_{\mathbf{u}}$.*

Proof Start with the definition of the empirical Rademacher complexity:

$$\hat{R}(\mathcal{H}^V_{E,D}) = \mathbf{E}_\sigma\left[\sup_{\mathbf{w}:\frac{1}{2}(\bar{D}\mathbf{w}^\top\mathbf{w}+D(\mathbf{w}^\top\mathbf{v}_i)^2)\leq E}\left|\frac{2}{n}\sum_{i=1}^n\sigma_i(\mathbf{w}^\top\mathbf{x}_i)\right|\right].$$

Consider the supremum inside the expectation. Depending on the sign of the term inside $|\cdot|$, the above corresponds to either a maximization or a minimization. Without loss of generality, consider the case of maximization. When a minimization is involved, the value of the objective still remains the same. The supremum is recovered by solving the following optimization problem:

$$\max_{\mathbf{w}}\ \mathbf{w}^\top\sum_{i=1}^n\sigma_i\mathbf{x}_i \quad \text{s.t.} \qquad \frac{1}{2}(\bar{D}\mathbf{w}^\top\mathbf{w} + D(\mathbf{w}^\top\mathbf{v}_i)^2) \leq E \qquad \forall 1\leq i\leq n_v. \tag{3.1}$$

Using Lagrange multipliers $\lambda_1 \geq 0, \ldots \lambda_{n_v} \geq 0$, the Lagrangian of (3.1) is: $\mathcal{L}(\mathbf{w},\lambda) = -\mathbf{w}^\top\sum_{i=1}^n\sigma_i\mathbf{x}_i + \sum_{i=1}^{n_v}\lambda_i\left(\frac{1}{2}\left(\bar{D}\mathbf{w}^\top\mathbf{w} + D(\mathbf{w}^\top\mathbf{v}_i)^2\right) - E\right)$. Differentiating this with respect to the primal variable $\mathbf{w}$ and equating it to zero gives: $\mathbf{w} = \mathbf{\Sigma}^{-1}_{\lambda,D}\sum_{i=1}^n\sigma_i\mathbf{x}_i$, where $\mathbf{\Sigma}_{\lambda,D} := \bar{D}\sum_{i=1}^{n_v}\lambda_i\mathbf{I} + D\sum_{i=1}^{n_v}\lambda_i\mathbf{v}_i\mathbf{v}_i^\top$. Substituting this $\mathbf{w}$ in $\mathcal{L}(\mathbf{w},\lambda)$ gives the dual of (3.1):

$$\min_{\lambda\geq 0}\ \frac{1}{2}\sum_{i=1}^n\sigma_i\mathbf{x}_i^\top\mathbf{\Sigma}^{-1}_{\lambda,D}\sum_{j=1}^n\sigma_j\mathbf{x}_j + E\sum_{i=1}^{n_v}\lambda_i.$$

[4] $T_2(\mathcal{A},\mathcal{B})$ has been defined on generic sets. When an already defined set, such as $\mathbf{V}$ (with a known number n_v of elements) is an argument to T_2, λ will be subscripted with i or j.

This permits the following upper bound on the empirical Rademacher complexity since the primal and the dual objectives are equal at the optimum:

$$\begin{aligned}\hat{R}(\mathcal{H}^V_{E,D}) &= \frac{2}{n}\mathbf{E}_\sigma\left[\min_{\lambda\geq 0}\ \frac{1}{2}\sum_{i=1}^{n}\sigma_i\mathbf{x}_i^\top\boldsymbol{\Sigma}_{\lambda,D}^{-1}\sum_{j=1}^{n}\sigma_j\mathbf{x}_j + E\sum_{i=1}^{n_v}\lambda_i\right]\\ &\leq \min_{\lambda\geq 0}\ \frac{2}{n}\mathbf{E}_\sigma\left[\frac{1}{2}\sum_{i=1}^{n}\sigma_i\mathbf{x}_i^\top\boldsymbol{\Sigma}_{\lambda,D}^{-1}\sum_{j=1}^{n}\sigma_j\mathbf{x}_j + E\sum_{i=1}^{n_v}\lambda_i\right]\\ &\leq \min_{\lambda\geq 0}\ \frac{1}{n}\sum_{i=1}^{n}\mathbf{x}_i^\top\boldsymbol{\Sigma}_{\lambda,D}^{-1}\mathbf{x}_i + \frac{2}{n}E\sum_{i=1}^{n_v}\lambda_i = T_2(\mathbf{V},\mathbf{S}).\end{aligned}$$

On line one, the expectation is over the minimizers over λ; this is less than first taking the expectation and then minimizing over λ in line two. Then, simply recycle the arguments used in Theorem 9 to handle the expectation over σ. ■

Theorem 11 $\hat{R}(\mathcal{G}^U_{B,D}) \leq T_1(\mathbf{U},\mathbf{S})$*, where for any training set* $\mathcal{B}$ *and landmark set* $\mathcal{A}$*,* $T_1(\mathcal{A},\mathcal{B}) := \frac{2\sqrt{2B}}{|\mathcal{B}|}\left(\sum_{\mathbf{x}\in\mathcal{B}}\mathbf{x}^\top\left(\bar{D}\mathbf{I} + \frac{D}{|\mathcal{A}|}\sum_{\mathbf{u}\in\mathcal{A}}\mathbf{u}\mathbf{u}^\top\right)^{-1}\mathbf{x}\right)^{\frac{1}{2}}$.

Proof The proof is similar to the one for Theorem 10. ■

Thus, the empirical Rademacher complexities of the function classes of interest are bounded using the functions T_0, $T_1(\mathbf{U},\mathbf{S})$ and $T_2(\mathbf{V},\mathbf{S})$. For both $\mathcal{F}_E$ and $\mathcal{G}^U_{E,D}$, the empirical Rademacher complexity is bounded by a closed-form expression. For $\mathcal{H}^V_{E,D}$, optimizing over the Lagrange multipliers (i.e. the λ's) can further reduce the upper bound on empirical Rademacher complexity. This can yield advantages over both $\mathcal{F}_E$ and $\mathcal{G}^U_{E,D}$ in many situations and the overall shape of $\boldsymbol{\Sigma}_{\lambda,D}$ plays a key role in determining the overall bound; this will be discussed in Section 3.5. Note that the upper bound $T_2(\mathbf{V},\mathbf{S})$ is not a closed-form expression in general but can be evaluated in polynomial time using semi-definite programming by invoking Schur's complement lemma as shown by [Boyd and Vandenberghe, 2003].

3.2.2 From empirical to true Rademacher complexity

By definition 8, the empirical Rademacher complexity of a function class is dependent on the data sample, $\mathbf{S}$. In many cases, it is not possible to give exact expressions for

the Rademacher complexity since the underlying distribution over the data is unknown. However, it is possible to give probabilistic upper bounds on the Rademacher complexity. Since the Rademacher complexity is the expectation of its empirical estimate over the data, by a straightforward application of McDiarmid's inequality (Appendix A.1), it is possible to show the following:

Lemma 12 *Fix $\delta \in (0,1)$. With probability at least $1-\delta$ over draws of the samples $\mathbf{S}$ the following holds for any function class $\mathcal{F}$:*

$$R(\mathcal{F}) \leq \hat{R}(\mathcal{F}) + 2\sqrt{\frac{\ln(2/\delta)}{2n}} \tag{3.2}$$

and,

$$\hat{R}(\mathcal{F}) \leq R(\mathcal{F}) + 2\sqrt{\frac{\ln(2/\delta)}{2n}}. \tag{3.3}$$

At this point, the motivation for introducing the landmark sets $\mathbf{U}$ and $\mathbf{V}$ becomes clear. The inequalities (3.2) and (3.3) do not hold when the function class $\mathcal{F}$ is dependent on the set $\mathbf{S}$. Specifically, using the sample $\mathbf{S}$ instead of the landmarks breaks the required *iid* assumptions in the derivation of (3.2) and (3.3). Thus neither Lemma 12, nor any of the results in Section 3.3 are sound for the function classes $\mathcal{G}^S_{B,D}$ and $\mathcal{H}^S_{E,D}$.

3.3 Generalization bounds

This section presents generalization bounds for the three different function classes. The derivation largely follows the approach of [Shawe-Taylor and Cristianini, 2004] and, therefore, several details will be omitted in this chapter. Recall the theorem from [Shawe-Taylor and Cristianini, 2004] that leverages the empirical Rademacher complexity to provide a generalization bound.

Theorem 13 *Let $\mathcal{F}$ be a class of functions mapping Z to $[0,1]$; let $\{\mathbf{z}_1, \ldots, \mathbf{z}_n\}$ be drawn from the domain Z independently and identically distributed (*iid*) according to a probability distribution $\mathcal{D}$. Then, for any fixed $\delta \in (0,1)$, the following bound holds for any $f \in \mathcal{F}$ with probability at least $1-\delta$ over random draws of a set of examples of size n:*

$$\mathbf{E}_{\mathcal{D}}[f(\mathbf{z})] \leq \hat{\mathbf{E}}[f(\mathbf{z})] + \hat{R}(\mathcal{F}) + 3\sqrt{\frac{\ln(2/\delta)}{2n}}. \tag{3.4}$$

Similarly, under the same conditions as above, with probability at least $1-\delta$*,*

$$\hat{\mathbf{E}}[f(\mathbf{z})] \leq \mathbf{E}_{\mathcal{D}}[f(\mathbf{z})] + \hat{R}(\mathcal{F}) + 3\sqrt{\frac{\ln(2/\delta)}{2n}}. \tag{3.5}$$

Inequality (3.4) can be found in [Shawe-Taylor and Cristianini, 2004] and inequality (3.5) is obtained by a simple modification of the proof in [Shawe-Taylor and Cristianini, 2004]. The following theorem, found in [Shawe-Taylor and Cristianini, 2004], gives a probabilistic upper bound on the future error rate based on the empirical error and the function class complexity.

Theorem 14 *Fix* $\gamma > 0$*. Let* $\mathcal{F}$ *be the class of functions from* $\mathbb{R}^m \times \{\pm 1\} \to \mathbb{R}$ *given by* $f(\mathbf{x}, y) = -yg(\mathbf{x})$*. Let* $\{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\}$ *be drawn* iid *from a probability distribution* $\mathcal{D}$*. Then, with probability at least* $1-\delta$ *over the samples of size* n*, the following bound holds:*

$$\mathbf{P}_{\mathcal{D}}[y \neq \text{sign}(g(\mathbf{x}))] \leq \frac{1}{n\gamma}\sum_{i=1}^{n}\xi_i + \frac{2}{\gamma}\hat{R}(\mathcal{F}) + 3\sqrt{\frac{\ln(2/\delta)}{2n}}, \tag{3.6}$$

where $\xi_i = \max(0, 1 - y_i g(\mathbf{x}_i))$ *are the so-called slack variables.*

The upper bounds that were derived in Section 3.2, namely: T_0, $T_1(\mathbf{U}, \mathbf{S})$ and $T_2(\mathbf{V}, \mathbf{S})$ can now be inserted into (3.6) to give generalization bounds for each class of interest. However, a caveat remains since a separate set of landmark data was necessary to provide such generalization bounds. The next section provides steps to eliminate the landmark data set from the bound.

3.4 Stating the bounds independently of landmarks

Note that the original function classes were defined using landmark examples. However, it is possible to eliminate these and state the generalization bounds independent of the landmark examples on function classes defined on the training data. Landmarks are eliminated from the generalization bounds in two steps. First, the empirical Rademacher complexities are shown to be concentrated and, second, the function classes defined using landmarks are shown to be supersets of the original function classes. One mild and standard assumption will be necessary, namely, that all examples from the distribution $\mathbf{P}([\mathbf{x}])$ have a norm bounded above by R with probability one.

3.4.1 Concentration of empirical Rademacher complexity

Recall the upper bound $T_1(\mathbf{U}, \mathbf{S})$ that was derived in Theorem 11. The following bounds show that these quantities are concentrated.

Theorem 15

With probability at least $1 - \delta$,

$$T_1(\mathbf{U}, \mathbf{S}) \leq \mathbf{E}_{\mathbf{U}}[T_1(\mathbf{U}, \mathbf{S})] + \mathcal{O}\left(\frac{1}{\sqrt{n}\sqrt{tr(\mathbf{K})}}\right),$$

$$T_2(\mathbf{V}, \mathbf{S}) \leq \mathbf{E}_{\mathbf{V}}[T_1(\mathbf{V}, \mathbf{S})] + \mathcal{O}\left(\frac{1}{\sqrt{n_v}\sqrt{tr(\mathbf{K})}}\right).$$

Proof McDiarmid's inequality from Appendix A.1 can be applied to $T_1(\mathbf{U}, \mathbf{S})$ since it is possible to compute Lipschitz constants $c_1, c_2, \ldots, c_n$ that correspond to each input of the function. These Lipschitz constants all share the same value c which is derived in Appendix A.2. With this Lipschitz constant, McDiarmid's inequality (A.1) is directly applicable and yields: $\mathbf{P}[T_1(\mathbf{U}, \mathbf{S}) - \mathbf{E}_{\mathbf{U}}[T_1(\mathbf{U}, \mathbf{S})] \geq \epsilon] \leq \exp\left(-2\epsilon^2/(nc^2)\right)$ Setting the upper bound on probability to δ, the following inequality holds with probability at least $1 - \delta$:

$$T_1(\mathbf{U}, \mathbf{S}) \leq \mathbf{E}_{\mathbf{U}}[T_1(\mathbf{U}, \mathbf{S})] + \frac{2\sqrt{\ln(1/\delta)E}}{\bar{D}\sqrt{n}}\left(\sqrt{\sum_{i=1}^{n}\mathbf{x}_i^\top\mathbf{x}_i} - \sqrt{\sum_{i=1}^{n}\mathbf{x}_i^\top\mathbf{x}_i - \frac{DR^2\mu_{max}}{n\bar{D} + DR^2}}\right). \tag{3.7}$$

The second term above is:

$$\begin{aligned}
&\frac{2\sqrt{\ln(1/\delta)E}}{\bar{D}\sqrt{n}}\left(\sqrt{\sum_{i=1}^{n}\mathbf{x}_i^\top\mathbf{x}_i} - \sqrt{\sum_{i=1}^{n}\mathbf{x}_i^\top\mathbf{x}_i - \frac{DR^2\mu_{max}}{n\bar{D} + DR^2}}\right) \\
&= \frac{2\sqrt{\ln(1/\delta)E}}{\bar{D}\sqrt{n}} \frac{DR^2\mu_{max}/(n\bar{D} + DR^2)}{\sqrt{\sum_{i=1}^{n}\mathbf{x}_i^\top\mathbf{x}_i} + \sqrt{\sum_{i=1}^{n}\mathbf{x}_i^\top\mathbf{x}_i - \frac{DR^2\mu_{max}}{n\bar{D}+DR^2}}} \\
&\leq \frac{2\sqrt{\ln(1/\delta)E}}{\bar{D}\sqrt{n}} \frac{DR^2\mu_{max}/(n\bar{D} + DR^2)}{\sqrt{\sum_{i=1}^{n}\mathbf{x}_i^\top\mathbf{x}_i}} \\
&\leq \frac{2\sqrt{\ln(1/\delta)E}}{\bar{D}\sqrt{n}} \frac{DR^4 n}{(n\bar{D} + DR^2)\sqrt{\sum_{i=1}^{n}\mathbf{x}_i^\top\mathbf{x}_i}} \\
&\leq \frac{2\sqrt{\ln(1/\delta)E}}{\bar{D}\sqrt{n}} \frac{DR^4 n}{(n\bar{D})\sqrt{\sum_{i=1}^{n}\mathbf{x}_i^\top\mathbf{x}_i}} = \mathcal{O}\left(\frac{1}{\sqrt{n}\sqrt{\mathrm{tr}(\mathbf{K})}}\right).
\end{aligned}$$

Here, $\mu_{max} \leq nR^2$ is the largest eigenvalue of the Gram matrix $\mathbf{K}$. The big oh notation refers to the asymptotic behavior in n. Note that $\text{tr}(\mathbf{K})$ also grows with n. Thus, asymptotically, the above term is better than $\mathcal{O}(1/\sqrt{n})$ which is the behavior of (3.6). So, from (3.7), with probability at least $1-\delta$: $T_1(\mathbf{U},\mathbf{S}) \leq \mathbf{E}_{\mathbf{U}}[T_1(\mathbf{U},\mathbf{S})] + \mathcal{O}\left(1/\sqrt{n\ \text{tr}(\mathbf{K})}\right)$.

The proof for the second claim is similar since $T_2(\mathbf{V},\mathbf{S})$ has the same Lipschitz constants (Appendix A.2). The only difference is in the number of elements in $\mathbf{V}$ which is reflected in the bound. ■

3.4.2 Function class inclusion

At this point, using Equation 3.6 and Theorem 15, it is possible to state bounds that hold for functions in $\mathcal{G}^U_{B,D}$ and $\mathcal{H}^U_{B,D}$ but that are independent of $\mathbf{U}$ and $\mathbf{V}$ otherwise. However, the aim is to state uniform convergence bounds for functions in $\mathcal{G}^S_{B,D}$ and $\mathcal{H}^S_{B,D}$. This is achieved by showing the latter two sets are subsets of the former two with high probability. It is not enough to show that each element of one set is inside the other. Since uniform bounds are required for the initial function classes, one has to prove set-inclusion results[5].

Theorem 16 *For $B = E + \epsilon$ where $\epsilon = \mathcal{O}\left(\frac{1}{\sqrt{n}}\right)$, with probability at least $1-2\delta$ $\mathcal{G}^S_{E,D} \subseteq \mathcal{G}^U_{B,D}$.*

Proof First, note that $\mathcal{G}^S_{E,D} \subseteq \mathcal{F}_{E/\bar{D}}$. Thus, $\mathcal{F}_{E/\bar{D}}$ is a bigger class of functions than $\mathcal{G}^S_{E,D}$. Moreover, $\mathcal{F}_{E/\bar{D}}$ is not dependent on data. Now, consider $\frac{\bar{D}}{2}\mathbf{w}^\top\mathbf{w} + \frac{D}{2}(\mathbf{w}^\top\mathbf{x})^2$ where $\mathbf{w} \in \mathcal{F}_{E/\bar{D}}$. For $||\mathbf{x}|| \leq R^2$, the Cauchy-Schwarz inequality yields $\sup_{\mathbf{w}\in\mathcal{F}_{E/\bar{D}}} \frac{\bar{D}}{2}\mathbf{w}^\top\mathbf{w} + \frac{D}{2}(\mathbf{w}^\top\mathbf{x})^2 \leq \kappa$ where $\kappa = E/2 + DER^2/(2\bar{D})$. Now, define the function $h^{\mathbf{w}} : \mathcal{R}^m \to [0,1]$, as : $h^{\mathbf{w}}(\mathbf{x}) = (\frac{\bar{D}}{2}\mathbf{w}^\top\mathbf{w} + \frac{D}{2}(\mathbf{w}^\top\mathbf{x})^2)/\kappa$. Since the sets $\mathbf{S}$ and $\mathbf{U}$ are drawn *iid* from the distribution $\mathcal{D}_x$, it is now possible to apply (3.4) and (3.5) for any $\mathbf{w} \in \mathcal{F}_{E/\bar{D}}$. Applying (3.5) to $h^{\mathbf{w}}(\cdot)$ on $\mathbf{S}$, $\forall \mathbf{w} \in \mathcal{F}_{E/\bar{D}}$, with probability at least $1-\delta$, the following inequality holds:

$$\mathbf{E}_{\mathcal{D}_x}[h^{\mathbf{w}}(\mathbf{x})] \leq \frac{1}{n}\sum_{i=1}^{n} h^{\mathbf{w}}(\mathbf{x}_i) + 2\sqrt{\frac{2E}{n\bar{D}}}\sqrt{\frac{1}{n}\text{tr}(\mathbf{K})} + 3\sqrt{\frac{\ln(2/\delta)}{2n}}, \tag{3.8}$$

[5] The function classes will also be treated as sets of parameters $\mathbf{w}$ without introducing additional notation.

where the value of $\hat{R}(\mathcal{F}_{E/\bar{D}})$ has been obtained from Theorem 9. The expectation is over the draw of $\mathbf{S}$. Similarly, applying (3.4) to $h^{\mathbf{w}}(\cdot)$ on $\mathbf{U}$, with probability at least $1-\delta$, $\forall \mathbf{w} \in \mathcal{F}_{E/\bar{D}}$, the following inequality holds:

$$\frac{1}{n}\sum_{i=1}^{n} h^{\mathbf{w}}(\mathbf{u}_i) \leq \mathbf{E}_{\mathcal{D}_x}[h^{\mathbf{w}}(\mathbf{u})] + 2\sqrt{\frac{2E}{n\bar{D}}}\sqrt{\frac{1}{n}\mathrm{tr}(\mathbf{K}_u)} + 3\sqrt{\frac{\ln(2/\delta)}{2n}} \tag{3.9}$$

where $\mathbf{K}_u$ is the Gram matrix of the landmark examples in $\mathbf{U}$. Using the fact that expectations in (3.8) and (3.9) are the same, $\mathrm{tr}(\mathbf{K}_u) \leq nR^2$, and the union bound, the following inequality holds $\forall \mathbf{w} \in \mathcal{F}_{E/\bar{D}}$ with probability at least $1-2\delta$:

$$\frac{1}{n}\sum_{i=1}^{n} h^{\mathbf{w}}(\mathbf{u}_i) \leq \frac{1}{n}\sum_{i=1}^{n} h^{\mathbf{w}}(\mathbf{x}_i) + 4R\sqrt{\frac{2E}{n\bar{D}}} + 6\sqrt{\frac{\ln(2/\delta)}{2n}}.$$

Using the definition of $h^{\mathbf{w}}(\cdot)$, with probability at least $1-2\delta$, $\forall \mathbf{w} \in \mathcal{F}_{E/\bar{D}}$,

$$\frac{\bar{D}}{2}\mathbf{w}^\top\mathbf{w} + \frac{D}{2n}\sum_{i=1}^{n}(\mathbf{w}^\top\mathbf{u}_i)^2 \leq \frac{\bar{D}}{2}\mathbf{w}^\top\mathbf{w} + \frac{D}{2n}\sum_{i=1}^{n}(\mathbf{w}^\top\mathbf{x}_i)^2 + \mathcal{O}\left(\frac{1}{\sqrt{n}}\right).$$

Now, suppose, $\frac{\bar{D}}{2}\mathbf{w}^\top\mathbf{w} + \frac{D}{2n}\sum_{i=1}^{n}(\mathbf{w}^\top\mathbf{x}_i)^2 \leq E$, which describes the function class $\mathcal{G}^S_{E,D}$. If B is chosen to be $E+\epsilon$ where $\epsilon = \mathcal{O}(\frac{1}{\sqrt{n}})$, then, $\forall \mathbf{w} \in \mathcal{F}_{E/\bar{D}}$, with probability at least $1-2\delta$, $\mathbf{w}^\top\mathbf{w} + \frac{D}{2n}\sum_{i=1}^{n}(\mathbf{w}^\top\mathbf{u}_i)^2 \leq B$. Since $\mathcal{F}_{E/\bar{D}}$ is a superset of $\mathcal{G}^S_{E,D}$, with probability at least $1-2\delta$, $\mathcal{G}^S_{E,D} \subseteq \mathcal{G}^U_{E,D}$. ■

Theorem 17 *For $n_v = \mathcal{O}(\sqrt{n})$, with probability at least $1-2\delta$, $\mathcal{H}^S_{E,D} \subseteq \mathcal{H}^V_{E,D}$.*

Proof First define the function, $g^{\mathbf{w}} : \mathbb{R}^m \to \mathbb{R}$, as $g^{\mathbf{w}}(\mathbf{v}) = \frac{\bar{D}}{2}\mathbf{w}^\top\mathbf{w} + \frac{D}{2}(\mathbf{w}^\top\mathbf{v})^2$. Define the indicator random variable $\mathbf{I}_{[g^{\mathbf{w}}(\mathbf{v})>E]}$ which has a value 1 if $g^{\mathbf{w}}(\mathbf{v}) > E$ and a value 0 otherwise. By definition, $\forall \mathbf{w} \in \mathcal{H}^S_{E,D}$, $\forall \mathbf{x}_i \in \mathbf{S}$, $\mathbf{I}_{[g^{\mathbf{w}}(\mathbf{x}_i)>E]} = 0$. Similarly, $\forall \mathbf{w} \in \mathcal{H}^V_{E,D}$, $\forall \mathbf{v}_i \in \mathbf{V}$, $\mathbf{I}_{[g^{\mathbf{w}}(\mathbf{v}_i)>E]} = 0$. As before, consider a larger class of functions that is independent of $\mathbf{S}$, namely, $\mathcal{F}_{E/\bar{D}}$. For an *iid* sample $\mathbf{S}$ from the distribution $\mathcal{D}_x$, applying (3.4) to the indicator random variables $\mathbf{I}_{[g^{\mathbf{w}}(\mathbf{x})>E]}$ on the set $\mathbf{S}$, with probability at least $1-\delta$,

$$\mathbf{E}_{\mathcal{D}_x}[\mathbf{I}_{[g^{\mathbf{w}}(\mathbf{x})>E]}] \leq \frac{1}{n}\sum_{i=1}^{n}\mathbf{I}_{[g^{\mathbf{w}}(\mathbf{x}_i)>E]} + 2\sqrt{\frac{2E}{n\bar{D}}}\sqrt{\frac{1}{n}\mathrm{tr}(\mathbf{K})} + 3\sqrt{\frac{\ln(2/\delta)}{2n}}. \tag{3.10}$$

Similarly, applying (3.5) on the set $\mathbf{V}$, with probability at least $1-\delta$,

$$\frac{1}{n_v}\sum_{i=1}^{n_v}\mathbf{I}_{[g^{\mathbf{w}}(\mathbf{v}_i)>E]} \leq \mathbf{E}_{\mathcal{D}_x}[\mathbf{I}_{[g^{\mathbf{w}}(\mathbf{x})>E]}] + 2\sqrt{\frac{2E}{n\bar{D}}}\sqrt{\frac{1}{n_v}\mathrm{tr}(\mathbf{K}_v)} + 3\sqrt{\frac{\ln(2/\delta)}{2n_v}}. \tag{3.11}$$

Performing a union bound on (3.10) and (3.11), using the fact that $\mathrm{tr}(\mathbf{K}) \leq nR^2$ and $\mathrm{tr}(\mathbf{K}_v) \leq n_v R^2$ with probability at least $1-2\delta$, $\forall \mathbf{w} \in \mathcal{F}_{E/\bar{D}}$,

$$\frac{1}{n_v}\sum_{i=1}^{n_v}\mathbf{I}_{[g^{\mathbf{w}}(\mathbf{v}_i)>E]} - \frac{1}{n}\sum_{i=1}^{n}\mathbf{I}_{[g^{\mathbf{w}}(\mathbf{x}_i)>E]} \leq 4R\sqrt{\frac{2E}{n\bar{D}}} + 3\sqrt{\frac{\ln(2/\delta)}{2}}\left(\frac{1}{\sqrt{n}}+\frac{1}{\sqrt{n_v}}\right). \tag{3.12}$$

Equating the right hand side of the above inequality to $\frac{1}{n_v}$, the above inequality can be written more succinctly as:

$$\begin{aligned}&\mathbf{P}\left[\exists \mathbf{w} \in \mathcal{F}_{E/\bar{D}} \frac{1}{n_v}\sum_{i=1}^{n_v}\mathbf{I}_{[g^{\mathbf{w}}(\mathbf{v}_i)>E]} - \frac{1}{n}\sum_{i=1}^{n}\mathbf{I}_{[g^{\mathbf{w}}(\mathbf{x}_i)>E]}) \geq \frac{1}{n_v}\right] \\ \leq & 2\exp\left(-\frac{2}{9}\left(\frac{1}{n_v} - 4R\sqrt{\frac{2E}{n\bar{D}}}\right)^2 \Big/ \left(\frac{1}{\sqrt{n}}+\frac{1}{\sqrt{n_v}}\right)^2\right)\end{aligned}$$

The left hand side of the equation above is the probability that there exists a $\mathbf{w}$ such that the difference in the fraction of the number of examples that fall outside $\frac{\bar{D}}{2}\mathbf{w}^\top\mathbf{w} + \frac{D}{2}(\mathbf{w}^\top\mathbf{x})^2 \leq E$ over the random draw of the sets $\mathbf{S}$ and $\mathbf{V}$ is at least $\frac{1}{n_v}$. Thus, it gives an upper bound on the probability that $\mathcal{H}^S_{E,D}$ is contained in $\mathcal{H}^V_{E,D}$. This is because, if there is a $\mathbf{w} \in \mathcal{H}^S_{E,D}$ that is not in $\mathcal{H}^V_{E,D}$, for such a $\mathbf{w}$, $\frac{1}{n_v}\sum_{i=1}^{n_v}\mathbf{I}_{[g^{\mathbf{w}}(\mathbf{v}_i)>E]} > \frac{1}{n_v}$ and $\frac{1}{n}\sum_{i=1}^{n}\mathbf{I}_{[g^{\mathbf{w}}(\mathbf{x}_i)>E]} = 0$. Thus, equating the right hand side of (3.12) to $\frac{1}{n_v}$ and solving for n_v, the result follows. Both an exact value and the asymptotic behavior of n_v are derived in Appendix A.3. ■

It is straightforward to write the generalization bounds of Section 3.3 only in terms of $\mathbf{S}$, completely eliminating the landmark set $\mathbf{U}$ from the results in this section. However, the resulting bounds now have additional factors which further loosen them. In spite of this, in principle, using a landmark set and compensating with McDiarmid's inequality can overcome the difficulties associated with a data-dependent hypothesis class and provide important generalization guarantees. In summary, the following overall bounds can now be provided for the function classes $\mathcal{F}_E$, $\mathcal{H}^S_{E,D}$ and $\mathcal{G}^S_{E,D}$. This result is obtained from a union bound of Theorem 14, Theorem 15, Theorem 16 and Theorem 17.

Theorem 18 *Fix $\gamma > 0$ and let $\{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\}$ be drawn* iid *from a probability distribution $\mathcal{D}$ where $\|x\| \leq R$.*

i) For any g from the function class $\mathcal{F}_E$, the following holds with probability at least $1 - \delta$,

$$\mathbf{P}_{\mathcal{D}}[y \neq \operatorname{sign}(g(\mathbf{x}))] \leq \frac{1}{\gamma n} \sum_{i=1}^{n} \xi_i + 3\sqrt{\frac{\ln(2/\delta)}{2n}} + \frac{4\sqrt{2E}}{n\gamma} \sqrt{tr(\mathbf{K})}. \tag{3.13}$$

ii) For any g from the function class $\mathcal{H}^S_{E,D}$, the following inequality (a solution of a semi-definite program) holds for $n_v = \mathcal{O}(\sqrt{n})$ with probability at least $1 - \delta$,

$$\begin{aligned}\mathbf{P}_{\mathcal{D}}[y \neq \operatorname{sign}(g(\mathbf{x}))] \leq \frac{1}{n\gamma} \sum_{i=1}^{n} \xi_i + 3\sqrt{\frac{\ln(8/\delta)}{2n}} + \mathcal{O}\left(\frac{1}{\sqrt{n_v}\sqrt{tr(\mathbf{K})}}\right) \\ + \frac{2}{\gamma}\mathbf{E}_{\mathbf{V}}\left(\min_{\lambda \geq 0} \frac{1}{n} \sum_{i=1}^{n} \mathbf{x}_i^\top \left(\bar{D} \sum_{j=1}^{n_v} \lambda_j \mathbf{I} + D \sum_{j=1}^{n_v} \lambda_j \mathbf{v}_j \mathbf{v}_j^\top\right)^{-1} \mathbf{x}_i + \frac{2E}{n} \sum_{i=1}^{n_v} \lambda_i\right).\end{aligned} \tag{3.14}$$

iii) Similarly, for any g from the function class $\mathcal{G}^S_{E,D}$, the following bound holds for $B = E + \mathcal{O}(\frac{1}{\sqrt{n}})$ with probability at least $1 - \delta$,

$$\begin{aligned}\mathbf{P}_{\mathcal{D}}[y \neq \operatorname{sign}(g(\mathbf{x}))] \leq \frac{1}{n\gamma} \sum_{i=1}^{n} \xi_i + 3\sqrt{\frac{\ln(8/\delta)}{2n}} + \mathcal{O}\left(\frac{1}{\sqrt{n}\sqrt{tr(\mathbf{K})}}\right) \\ + \frac{4\sqrt{2B}}{n\gamma}\mathbf{E}_{\mathbf{U}}\left(\sum_{i=1}^{n} \mathbf{x}_i^\top \left(\bar{D}\mathbf{I} + \frac{D}{n} \sum_{j=1}^{n} \mathbf{u}_j \mathbf{u}_j^\top\right)^{-1} \mathbf{x}_i\right)^{\frac{1}{2}},\end{aligned} \tag{3.15}$$

where $\xi_i = \max(0, \gamma - y_i g(\mathbf{x}_i))$ are the so-called slack variables.

3.5 Discussion of the bounds

Clearly, all the three bounds, namely (3.13), (3.14) and (3.15) in Theorem (18) have similar asymptotic behavior in n, so how do they differ? Simple, separable scenarios are considered in this section to examine these bounds (which will be referred to as the SVM bound, RMM bound and $\mathbf{\Sigma}$-SVM bound respectively). For the SVM bound, the quantity of interest is $4\frac{\sqrt{2E}}{n\gamma}\sqrt{\operatorname{tr}(\mathbf{K})}$ and, for the $\mathbf{\Sigma}$-SVM bound, the quantity of interest is $\frac{4\sqrt{2E}}{n\hat{\gamma}}\sqrt{\left(\sum_{i=1}^{n} \mathbf{x}_i^\top \left(\bar{D}\mathbf{I} + \frac{D}{n}\sum_{j=1}^{n} \mathbf{u}_j \mathbf{u}_j^\top\right)^{-1} \mathbf{x}_i\right)}$. Similarly, for the RMM bound, the quan-

tity of interest is:

$$\frac{2}{\hat{\gamma}}\left(\min_{\lambda\geq 0}\ \frac{1}{n}\sum_{i=1}^{n}\mathbf{x}_i^\top\left(\bar{D}\sum_{j=1}^{n_v}\lambda_j\mathbf{I}+D\sum_{j=1}^{n_v}\lambda_j\mathbf{v}_j\mathbf{v}_j^\top\right)^{-1}\mathbf{x}_i+\frac{2E}{n}\sum_{i=1}^{n_v}\lambda_i\right).$$

Here the expectations over $\mathbf{U}$ and $\mathbf{V}$ have been dropped for brevity; in fact, this is how these terms would have appeared without the concentration result (Theorem 15). Moreover, in the latter two cases, γ has been replaced by $\hat{\gamma}$ intentionally.

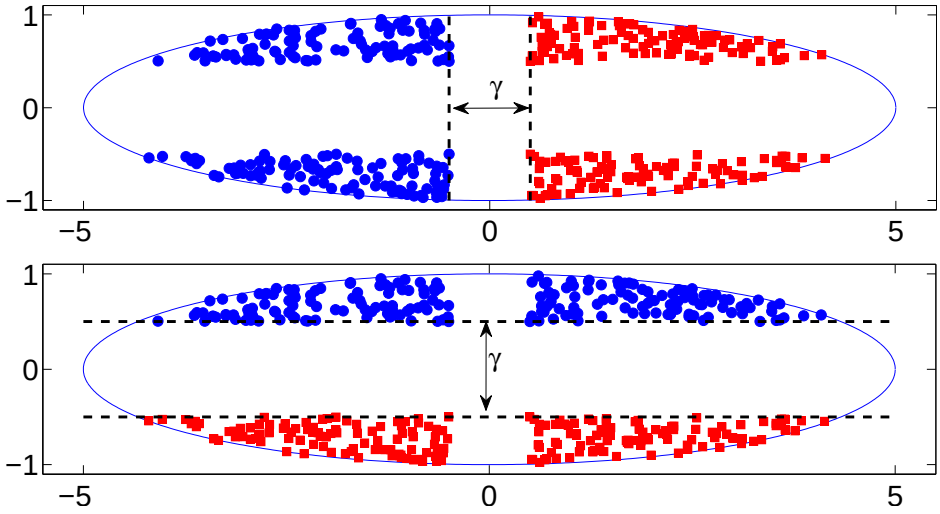

Figure 3.1: Two labellings of the same examples. Circles and squares denote the two classes (positive and negative). The top case is referred to as "toy example 1" and the bottom case is referred to as "toy example 2" in the sequel. The bound for the function class $\mathcal{F}_E$ does not distinguish between these two cases.

The differences between the three bounds will be illustrated with a toy example. In Figure 3.1, two different labellings of the same dataset are shown. The two different labellings of the data produce completely different classification boundaries. However, in both the cases, the absolute margin of separation γ remains the same. A similar synthetic setting was explored in the context of second order perceptron bounds [Cesa-Bianchi *et al.*, 2005].

The margin γ corresponding to the function class $\mathcal{F}$ is found by solving the following optimization problem:

$$\max_{\gamma,\mathbf{w}}\ \gamma,\quad \text{s.t. } y_i(\mathbf{w}^\top\mathbf{x}_i)\geq\gamma,\ \frac{1}{2}\mathbf{w}^\top\mathbf{w}\leq E.$$

	toy example 1	toy example 2
SVM bound	0.643	0.643
$\mathbf{\Sigma}$-SVM bound, D=0	0.643	0.643
$\mathbf{\Sigma}$-SVM bound, D=0.999	0.859	0.281
RMM bound, D=0	0.643	0.643
RMM bound, D=0.999	1.355	0.315

Table 3.1: The bound values for the two toy examples. The SVM bound does not distinguish between the two cases. By exploring D values, it is possible to obtain smaller bound values in both cases for $\mathbf{\Sigma}$-SVM and RMM ($D = 0$ in toy example 1 and D close to one in toy example 2).

This merely recovers the absolute margin γ which is shown in the figure. Similarly, for the function class $\mathcal{G}$, a margin $\hat{\gamma}$ is obtained by solving:

$$\max_{\gamma,\mathbf{w}} \ \gamma, \text{s.t. } y_i(\mathbf{w}^\top \mathbf{x}_i) \geq \gamma, \ \frac{1}{2}\mathbf{w}^\top \left(\bar{D}\mathbf{I} + \frac{D}{n}\sum_{j=1}^{n} \mathbf{x}_j \mathbf{x}_j^\top \right) \mathbf{w} \leq E.$$

Through a change of variable, $\mathbf{u} = \mathbf{\Sigma}^{\frac{1}{2}}\mathbf{w}$ where $\mathbf{\Sigma} = \left(\bar{D}\mathbf{I} + \frac{D}{n}\sum_{j=1}^{n} \mathbf{x}_j \mathbf{x}_j^\top \right)$ it is easy to see that the above optimization problem is equivalent to

$$\max_{\gamma,\mathbf{u}} \ \gamma, \text{ s.t. } y_i \mathbf{u}^\top \mathbf{\Sigma}^{-\frac{1}{2}} \mathbf{x}_i \geq \gamma, \ \frac{1}{2}\mathbf{u}^\top \mathbf{u} \leq E.$$

Thus, when a linear function is selected from the function class $\mathcal{G}^S_{D,E}$, the margin $\hat{\gamma}$ is estimated from a whitened version of the data. Similarly, for function class $\mathcal{H}^S_{E,D}$, the margin is estimated from a whitened version of the data where the whitening matrix is modified by Lagrange multipliers.

Thus, in the finite sample case, the bounds differ as demonstrated in the above synthetic problem. The bound for the function class $\mathcal{G}^S_{E,D}$ explores a whitening of the data. Suppose $D = 0.999$, the result is a whitening which evens out the spread of the data in all directions. On this whitened data set, the margin $\hat{\gamma}$ appears much larger in toy example 2 since it is large compared to the spread. This leads to an improvement in the $\mathbf{\Sigma}$-SVM bound over the usual SVM bound. While such differences could be compensated for by appropriate a priori normalization of features, this is not always an easy preprocessing.

Similarly, the RMM bound also considers a whitening of the data however, it shapes the whitening matrix adaptively by estimating λ. This gives further flexibility and rescales data not only along principal eigen-directions but in any direction where the margin is large relative to the spread of the data. By exploring D values, margin can be measured relative to the spread of the data rather than in the absolute sense. Since $\mathbf{\Sigma}$-SVM and RMM are strict generalizations of the SVM, through the use of a proper validation set, it is almost always possible to obtain improvements. The values of the bounds for the two toy examples are shown in Table 3.1.

Chapter 4

Structured Prediction

Traditionally, machine learning algorithms were designed to handle simple outputs such as a real valued target or a class label from k possible classes. With the application of machine learning tools in ambitious areas such as natural language processing, biology and computer vision, many machine learning techniques have been extended to provide complex objects as outputs. For example, in natural language processing, input examples are sentences while outputs are tags such as the role of the word or the part-of-speech the word represents in the sentence. It is difficult to apply traditional approaches to such problems, in particular, when the number of possible outputs can be exponentially large (as a function of the length of the output). For example, brute force enumeration of all the classification constraints implicated by a complex structured output space may be prohibitive in a traditional SVM setting. Recently, there has been extensive interest in the machine learning community for solving such problems involving complex outputs [Bakir *et al.*, 2006].

An example structured prediction problem is learning from sequences. Traditionally, hidden Markov models (HMM) were used in learning problems where output predictions are sequences. Discriminative methods have recently been brought to bear on sequence problems. Conditional random fields (CRFs) [Lafferty *et al.*, 2001] take a probabilistic approach to avoid constraint enumeration. Boosting and online frameworks [Altun *et al.*, 2003a] have also been proposed. Subsequently, hidden Markov support vector machines [Altun *et al.*, 2003b] and maximum margin Markov networks [Taskar *et al.*, 2004] produced improved accuracy by maximizing margin while mimicking HMM and Markov network-style

dependencies between inputs and outputs. Recently, a more general solution was proposed in the SVM framework for structured prediction problems [Tsochantaridis *et al.*, 2004; Tsochantaridis *et al.*, 2005]. The approach of [Tsochantaridis *et al.*, 2004; Tsochantaridis *et al.*, 2005] measures the margin in an absolute sense. Relative margin versions of these algorithms are developed in this chapter (these formulations were first proposed in [Shivaswamy and Jebara, 2009]). The organization of this chapter is as follows.

Structured support vector machines are first discussed in Section 4.1. Relative margin formulation for structured prediction is proposed in Section 4.2. A cutting plane algorithm and its analysis are provided in Section 4.3. Experimental evidence on label sequence learning and multi-class problems are presented in Section 4.4. A summary of this chapter is presented in Section 4.5.

4.1 Structured prediction with support vector machines

In structured prediction problems, a mapping $f : \mathcal{X} \to \mathcal{Y}$ from an input space to a discrete output space is estimated from training data. This mapping function is recovered by performing a maximization involving another augmented function $F : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$ such that:

$$f(\mathbf{x}) = \arg\max_{\mathbf{y} \in \mathcal{Y}} F(\mathbf{x}, \mathbf{y}; \mathbf{w}). \tag{4.1}$$

where $\mathbf{w}$ denotes the parameters of the function F.

As in most kernel based methods, the functional form is assumed to be linear,[1] *i.e.*, $F(\mathbf{x}, \mathbf{y}; \mathbf{w}) = \mathbf{w}^\top \boldsymbol{\psi}(\mathbf{x}, \mathbf{y})$, where $\boldsymbol{\psi}(\mathbf{x}, \mathbf{y})$ is a joint feature mapping for the pair $(\mathbf{x}, \mathbf{y})$. The specific form of this feature mapping depends on the application.

Let $(\mathbf{x}_i, \mathbf{y}_i)_{i=1}^n$ be the training dataset. Consider a pair $(\mathbf{x}_i, \mathbf{y}_i) \in \mathcal{X} \times \mathcal{Y}$ and consider the following constraint:

$$\mathbf{w}^\top \boldsymbol{\psi}(\mathbf{x}_i, \mathbf{y}_i) - \mathbf{w}^\top \boldsymbol{\psi}(\mathbf{x}_i, \mathbf{y}) \geq 1.$$

This constraint ensures that the real valued prediction F for $(\mathbf{x}_i, \mathbf{y}_i)$ is larger than the prediction for $(\mathbf{x}_i, \mathbf{y})$. In the SVM framework, parameters of the hyperplane $(\mathbf{w})$ are found by

[1] As before, merely the dot products between $\mathbf{w}$ and the joint feature maps are shown with the understanding that they can be replaced by a generalized inner product in a reproducing kernel Hilbert space.

maximizing the margin, while ensuring the above constraint holds and scores each training example i higher than all possible alternative labellings $\mathbf{y} \neq \mathbf{y}_i$. In particular, the structured SVM formulation[2] as proposed in [Tsochantaridis *et al.*, 2005] solves the following optimization problem to recover $\mathbf{w}$:

$$\begin{aligned} \min_{\mathbf{w}} \ & \frac{1}{2}\mathbf{w}^\top \mathbf{w} \\ \text{s.t. } & \mathbf{w}^\top \boldsymbol{\psi}(\mathbf{x}_i, \mathbf{y}_i) - \mathbf{w}^\top \boldsymbol{\psi}(\mathbf{x}_i, \mathbf{y}) \geq 1 \qquad \forall i \in N, \mathbf{y} \neq \mathbf{y}_i. \end{aligned} \tag{4.2}$$

For brevity, the set $\{1, 2, \ldots, n\}$ is denoted by N. It is customary to denote the quantity $\boldsymbol{\psi}(\mathbf{x}_i, \mathbf{y}_i) - \boldsymbol{\psi}(\mathbf{x}_i, \mathbf{y})$ by $\boldsymbol{\delta\psi}_i(\mathbf{y})$ and permit partial violations of the constraints in (4.2), by introducing so-called slack variables which yields the nonseparable structured prediction version of the SVM (referred to as StructSVM) as follows:

$$\begin{aligned} \min_{\mathbf{w},\xi} \ & \frac{1}{2}\mathbf{w}^\top \mathbf{w} + \frac{C}{n}\sum_{i=1}^{n} \xi_i \\ \text{s.t. } & \mathbf{w}^\top \boldsymbol{\delta\psi}_i(\mathbf{y}) \geq 1 - \xi_i \qquad \forall i \in N, \mathbf{y} \neq \mathbf{y}_i \\ & \xi_i \geq 0 \quad \forall i \in N, \end{aligned} \tag{4.3}$$

where $C > 0$ is a parameter that trades off between the margin and the slack variables. Recovering $\mathbf{w}$ with the above maximum-margin optimization problem and using $f(\mathbf{x})$ for prediction leads to state-of-the-art performance on many structured prediction benchmark tasks [Tsochantaridis *et al.*, 2005].

In chapter 2 relative margin machine was developed based on arguments from affine invariance and generalization. In this chapter, similar ideas are extended to structured prediction problems with formulation (4.3) as the starting point. In binary classification problems, the idea was to separate the two classes with a large (relative) margin. In structured prediction problems, the joint feature map of a training example with its correct label is separated with a large (relative) margin from the joint feature map of the same training example with any other labeling. In terms of the notations introduced earlier, StructSVM was shown to separate the joint feature map of $\mathbf{x}_i$ and its correct label $\mathbf{y}_i$ (*i.e.*, $\boldsymbol{\psi}(\mathbf{x}_i, \mathbf{y}_i)$)

[2]For brevity, the margin rescaling and slack rescaling extensions are omitted; they are straightforward extensions.

from the joint feature map of $\mathbf{x}_i$ with any other label $\mathbf{y} \in \mathcal{Y} \setminus \{\mathbf{y}_i\}$ (*i.e.,* $\boldsymbol{\psi}(\mathbf{x}_i, \mathbf{y})$) with a large margin. The key idea of the RMM in the binary classification case is to maximize the margin while bounding the spread of the projections. In the structured prediction case, the idea is to separate the "winner" $\mathbf{w}^\top \psi(\mathbf{x}_i, \mathbf{y}_i)$ from the "runner up" $\max_{\mathbf{y} \neq \mathbf{y}_i} \mathbf{w}^\top \psi(\mathbf{x}_i, \mathbf{y})$ with a large margin. Thus, the relative margin version of it would correspond to separating the winner from the runner up with a large margin with respect to the separation between the winner and the "worst contender" $\min_{\mathbf{y} \neq \mathbf{y}_i} \mathbf{w}^\top \psi(\mathbf{x}_i, \mathbf{y})$.

The cutting plane algorithm for StructSVM exploits the fact that the $\min \mathbf{w}^\top \psi(\mathbf{x}_i, \mathbf{y})$ over $\mathbf{y}$ is efficiently computable even if the output space $\mathcal{Y}$ is exponentially large. Bounding the separation between the winner and the worst contender would require max of $\mathbf{w}^\top \psi(\mathbf{x}_i, \mathbf{y})$ over $\mathbf{y}$ as well. In fact, in most structured spaces, only the minimum and maximum over $\mathcal{Y}$ of the function $\mathbf{w}^\top \psi(\mathbf{x}_i, \mathbf{y})$ are estimable efficiently while estimating the mean, median or variance of $\mathbf{w}^\top \psi(\mathbf{x}_i, \mathbf{y})$ over $\mathcal{Y}$ might be NP-hard or computationally prohibitive. This particular property is exploited to obtain an efficient algorithm for the structured prediction RMM.

4.2 Structured RMM

The basic idea of the RMM was to maximize the margin while limiting the spread of the projections. Such an idea can be naturally extended to structured prediction problems by maintaining the constraints from the formulation (4.3) while also incorporating an upper bound B on the difference in projections:

$$\begin{aligned} \min_{\mathbf{w},\xi} \ & \frac{1}{2}\mathbf{w}^\top \mathbf{w} + \frac{C}{n}\sum_{i=1}^{n} \xi_i & & (4.4) \\ \text{s.t. } & \mathbf{w}^\top \boldsymbol{\delta\psi}_i(\mathbf{y}) \geq 1 - \xi_i & & \forall i \in N, \mathbf{y} \neq \mathbf{y}_i \\ & \xi_i \geq 0 & & \forall i \in N \\ & -B \leq \mathbf{w}^\top \boldsymbol{\delta\psi}_i(\mathbf{y}) \leq B & & \forall i \in N, \forall \mathbf{y} \in \mathcal{Y}. \end{aligned}$$

The original structured prediction constraints in Equation (4.4) are referred to as the classification constraints while the added constraints are referred to as the bounding constraints. Here, $B > 1$ is an additional parameter in the new framework which trades off between mar-

gin maximization and spread minimization by bounding the projections. When $B = \infty$, the StructSVM method is exactly recovered. For other settings of B, the new formulation may produce solutions that improve generalization accuracy. Although the parameter B seems somewhat ad-hoc, experiments in Section 4.4 show that it can be set systematically as with other parameters in machine learning algorithms (e.g. the C value in StructSVM) and is easily adjusted using validation.

While the primal problem above is useful for deriving the algorithm, in practice, the dual of (4.4) is solved. The cutting plane algorithm is a standard tool for optimizing the dual and only kernel evaluations between the pairs $\boldsymbol{\delta\psi}_i(\mathbf{y})$ are required. The dual of (4.4) can be shown to be:

$$\begin{aligned}
\max_{\alpha,\beta,\beta^*} \quad & \sum_{i,\mathbf{y}\neq\mathbf{y}_i} \alpha_{(i\mathbf{y})} - B\sum_{i,\mathbf{y}} \beta_{(i\mathbf{y})} - B\sum_{i,\mathbf{y}} \beta^*_{(i\mathbf{y})} \\
& - \frac{1}{2}\left(\sum_{i,\mathbf{y}\neq\mathbf{y}_i} \alpha_{(i\mathbf{y})}\boldsymbol{\delta\psi}_i(\mathbf{y}) - \sum_{i,\mathbf{y}} \beta_{(i\mathbf{y})}\boldsymbol{\delta\psi}_i(\mathbf{y}) + \sum_{i,\mathbf{y}} \beta^*_{(i\mathbf{y})}\boldsymbol{\delta\psi}_i(\mathbf{y})\right)^\top \\
& \left(\sum_{j,\mathbf{y}\neq\mathbf{y}_j} \alpha_{(j\mathbf{y})}\boldsymbol{\delta\psi}_j(\mathbf{y}) - \sum_{j,\mathbf{y}} \beta_{(j\mathbf{y})}\boldsymbol{\delta\psi}_j(\mathbf{y}) + \sum_{j,\mathbf{y}} \beta^*_{(j\mathbf{y})}\boldsymbol{\delta\psi}_j(\mathbf{y})\right) \\
\text{s.t. } & 0 \leq \sum_{\mathbf{y}\neq\mathbf{y}_i} \alpha_{(i\mathbf{y})} \leq \frac{C}{n} \quad \forall i \in N, \\
& \alpha_{(i\mathbf{y})} \geq 0, \beta_{(i\mathbf{y})} \geq 0, \beta^*_{(i\mathbf{y})} \geq 0 \quad \forall i \in N,
\end{aligned} \tag{4.5}$$

where $\alpha_{(i\mathbf{y})}, \beta_{(i\mathbf{y})}$ and $\beta^*_{(i\mathbf{y})}$ are the Lagrange multipliers of the primal constraints. From the dual variables, $\mathbf{w}$ can be readily obtained as:

$$\mathbf{w} = \sum_{i,\mathbf{y}\neq\mathbf{y}_i} \alpha_{(i\mathbf{y})}\boldsymbol{\delta\psi}_i(\mathbf{y}) - \sum_{i,\mathbf{y}} \beta_{(i\mathbf{y})}\boldsymbol{\delta\psi}_i(\mathbf{y}) + \sum_{i,\mathbf{y}} \beta^*_{(i\mathbf{y})}\boldsymbol{\delta\psi}_i(\mathbf{y}).$$

4.3 Cutting Plane Algorithm

Similar to the algorithm for StructSVM, the cutting plane algorithm for StructRMM starts with an empty working set of primal constraints. The constraints in the primal formulation correspond to the variables in the dual formulation. Hence, the algorithms can be equivalently seen as starting with an empty working set of dual variables. At each step,

Algorithm 4.1 Cutting plane algorithm for StructRMM.

Require: $(\mathbf{x}_i, y_i)_{i=1}^n$, Parameters: $C, B, \epsilon, \epsilon_B$.

$S_i^1 \leftarrow \emptyset$, $S_i^2 \leftarrow \emptyset$, and $S_i^3 \leftarrow \emptyset$ forall $1 \leq i \leq n$

repeat

  for $i \leftarrow 1$ to n **do**

    $flag \leftarrow 0$

    $\tilde{\mathbf{y}}_1 \leftarrow \arg\max_{\mathbf{y}\in\mathcal{Y}} H(\mathbf{y}) := 1 - \boldsymbol{\delta\psi}_i(\mathbf{y})^\top \mathbf{w}$

    $\tilde{\mathbf{y}}_2 \leftarrow \arg\max_{\mathbf{y}\in\mathcal{Y}} G(\mathbf{y}) := \boldsymbol{\delta\psi}_i(\mathbf{y})^\top \mathbf{w}$

    $\tilde{\mathbf{y}}_3 \leftarrow \arg\min_{\mathbf{y}\in\mathcal{Y}} -G(\mathbf{y})$

    $\{$ where $\mathbf{w} = \sum_j \sum_{\mathbf{y}'\in S_j} \alpha_{(j\mathbf{y}')} \boldsymbol{\delta\psi}_j(\mathbf{y}') - \sum_j \sum_{\mathbf{y}'\in U_j} \beta_{(j\mathbf{y}')} \boldsymbol{\delta\psi}_j(\mathbf{y}') + \sum_j \sum_{\mathbf{y}'\in U_j} \beta^*_{(j\mathbf{y}')} \boldsymbol{\delta\psi}_j(\mathbf{y}')$ $\}$

    $\xi_i \leftarrow \max(0, \max_{\mathbf{y}\in S_i} H(\mathbf{y}))$

    $Violation_1 \leftarrow H(\tilde{\mathbf{y}}_1) - \xi_i - \epsilon$

    $Violation_2 \leftarrow G(\tilde{\mathbf{y}}_2) - B - \epsilon_B$

    $Violation_3 \leftarrow -G(\tilde{\mathbf{y}}_3) - B - \epsilon_B$

    $j \leftarrow \arg\max_{i\in\{1,2,3\}} Violation_i$

    if $Violation_j > 0$ **then**

      $S_i^j \leftarrow S_i^j \cup \{\tilde{\mathbf{y}}_j\}$, $flag \leftarrow 1$

    end if

    if $flag$ **equal to** 1 **then**

      Optimize the dual over S^1, S^2 and S^3, update α, β, β^*

    end if

  end for

until no variables are added to S_i^1, S_i^2 or S_i^3 for any i

most violated constraint for the i^{th} example is found. If there is any such violation, the corresponding dual variable is added to the working set. The algorithm then optimizes the dual objective over the variables in the working set. The steps involved in the cutting plane algorithm for StructRMM are shown in Algorithm 4.1.

Unlike the cutting plane algorithm for StructSVM, the algorithm for StructRMM re-

quires both the maximum and the minimum of $\mathbf{w}^\top \boldsymbol{\delta\psi}_i(\mathbf{y})$ over $\mathbf{y} \neq \mathbf{y}_i$. It is possible to efficiently find the maximum and the minimum structured prediction in many problems. For example, in label sequence learning problems, finding the above maximum (minimum) corresponds to finding the longest (shortest) paths in a graph induced by the emission and the transition probabilities. If there is an algorithm to find the longest path, the shortest path can be found merely by negating all weights. This is true for a variety of algorithms used in structured prediction such as Viterbi decoding, max-product belief propagation, maximum weight spanning tree estimation, and maximum weight matching.

As with the cutting plane algorithm for StructSVM, it is necessary to find the second maximum or the minimum possible label sequence to exclude the possibility of adding label $\mathbf{y}_i$ for the i^{th} example. This can be achieved efficiently [Chow and Schwartz, 1991] as well.

In label sequence learning problems, the most violated constraints are found using dynamic programming which corresponds to the Viterbi decoding algorithm. Therefore, the steps of Algorithm 4.1 that involve an arg max or an argmin over $\mathbf{y}$ (which resides in a large space $\mathcal{Y}$ of possible outputs) can be solved efficiently via fast decoding algorithms.

4.3.1 Runtime

First, recall the following lemma and the corollary from [Tsochantaridis *et al.*, 2005]:

Lemma 19 *Consider the following bounded (from above) concave objective in* $\boldsymbol{\gamma}$

$$\theta(\boldsymbol{\gamma}) = -\frac{1}{2}\boldsymbol{\gamma}^\top \mathbf{J}\boldsymbol{\gamma} + \mathbf{h}^\top \boldsymbol{\gamma},$$

where $\mathbf{J}$ *is a symmetric, positive semi-definite matrix. Assume that an optimization direction* $\boldsymbol{\eta}$ *and a starting point* $\boldsymbol{\gamma}^0$ *are given. Then, optimizing the objective* θ *in the given direction starting from* $\boldsymbol{\gamma}^0$ *increases the objective by:*

$$\frac{1}{2}\frac{\left(\nabla\theta(\boldsymbol{\gamma}^0)^\top \boldsymbol{\eta}\right)^2}{\boldsymbol{\eta}^\top \mathbf{J}\boldsymbol{\eta}} > 0.$$

Corollary 20 *For the special case of axis-aligned optimization direction* $\boldsymbol{\eta} = e_r$*, under the same conditions as above, the objective improves by*

$$\frac{1}{2\mathbf{J}_{rr}}\left(\frac{\partial\theta}{\partial\boldsymbol{\gamma}_r}\right)^2 > 0.$$

First, note that, if in the cutting plane algorithms, a classification constraint is the worst violator, the expression for improvements from [Tsochantaridis *et al.*, 2005] still hold in each step. It is given by,

$$\min\left\{\frac{C\epsilon}{2n}, \frac{\epsilon^2}{8R_i^2}\right\} \tag{4.6}$$

where $R_i = \max_{\mathbf{y}}\{\|\boldsymbol{\delta\psi}_i(\mathbf{y})\|\}$.

Note that the Lagrange multipliers corresponding to the bounding constraints in the primal StructRMM formulations can only be positive; there are no additional constraints on them. Thus, when the worst violations in the cutting plane algorithm are bounding constraints, we can simply apply Corollary 20. Thus, an expression can be derived for the improvement obtained by a step of the cutting plane method in Algorithm 4.1.

First, $\boldsymbol{\gamma}$ is identified with the variables α, β and β^* in (4.5). However, note that only a subset of all these variables is added when the dual is solved using the cutting plane algorithm. $\mathbf{J}$ is identified with the positive semi-definite kernel matrix whose ij^{th} term is formed from the kernel between $\boldsymbol{\delta}\psi_i(\mathbf{y})$ and $\boldsymbol{\delta}\psi_j(\mathbf{y})$ for two of the added constraints in the cutting plane algorithm.

Adding a dual variable in the cutting plane algorithm simply corresponds to optimizing over an axis parallel direction e_r. As stated before, it is only necessary to consider adding a dual variable β or β^* in (4.5) since the improvement in the objective is already known when a classification constraint is added.

Suppose, a dual variable $\beta_{(i\mathbf{y})}$ is added in the cutting plane algorithm to the set S^2, then:

$$\frac{\partial\theta}{\partial\beta_{(i\mathbf{y})}}(\boldsymbol{\gamma}^0) = -B + \mathbf{w}^{*\top}\boldsymbol{\delta\psi}_i(\mathbf{y}),$$

where $\mathbf{w}^*$ is the current solution of the cutting plane algorithm.

The fact that this dual variable was added to the working set in Step 15 to S^2 implies that

$$\mathbf{w}^{*\top}\boldsymbol{\delta\psi}_i(\mathbf{y}) > B + \epsilon_B,$$

since only violating constraints result in such additions to the working sets. As a consequence,

$$\frac{\partial\theta}{\partial\beta_{(i\mathbf{y})}}(\boldsymbol{\gamma}^0) > \epsilon_B.$$

Finally, $\mathbf{J}_{(i\mathbf{y})(i\mathbf{y})} = \boldsymbol{\delta\psi}_i(\mathbf{y})^\top \boldsymbol{\delta\psi}_i(\mathbf{y}) \leq R_i^2$.

It is now possible to insert both terms in Corollary 20 to get an expression for the minimum improvement achieved when an element is added to S^2. Thus:

$$\boldsymbol{\delta}\theta(\boldsymbol{\gamma}^0) \geq \frac{1}{2}\frac{\epsilon_B^2}{R_i^2}.$$

Exactly the same improvement is achieved when an element is added to S^3 in a similar way.

Once the improvement in the objective is known for both types of constraints, the overall guaranteed improvement is the minimum of the two possible improvements in each step. Thus, considering (4.6), the improvement in each step is:

$$\min\left\{\frac{1}{2}\frac{\epsilon_B^2}{R_i^2}, \frac{C\epsilon}{2n}, \frac{\epsilon^2}{8R_i^2}\right\}.$$

Finally, observe that the initial value of the dual objective is 0 when no variables are in the working sets. Since the dual objective is upper bounded by the primal objective, C is an upper bound. Therefore, it is possible to calculate the number of steps required for Algorithm 4.1 to terminate. The above arguments are summarized in the following theorem.

Theorem 21 *With $\bar{R} = \max_i R_i$, for any $\epsilon > 0$, $\epsilon_B > 0$, Algorithm 4.1 terminates after adding*

$$\max\left\{\frac{2\bar{R}^2 C}{\epsilon_B^2}, \frac{2n}{\epsilon}, \frac{8C\bar{R}^2}{\epsilon^2}\right\}$$

constraints to the working sets S^1, S^2 or S^3.

In practice, the implementation of Algorithm 4.1 can actually add all the top violating constraints of each type for each training example; this may further speedup the algorithm since jointly optimizing over several variables can give bigger improvements than optimizing over a single variable. Furthermore, the upper bound on the number of steps of the cutting plane method is very conservative and does not show any direct dependence of B. However, in practice, smaller B values require more time to converge. Nevertheless, while the theoretical bounds on convergence are helpful, practical experiments show that the proposed method brings improvement in accuracy.

4.4 Experiments

This section provides empirical justification for the proposed algorithm on label sequence learning and multi-class problems. [Altun *et al.*, 2003b] showed that the discriminative methods typically performed better than generative approaches on the label sequence learning problems. From the results from [Altun *et al.*, 2003b], it is clear that StructSVM improved over a regularized version of the CRF. Thus, results are obtained for StructSVM, StructRMM and a regularized CRF in label sequence learning problems. Primary comparisons are between StructSVM and StructRMM in the case of multi-class classification problems. In all the experiments, features are normalized to a *zero-one box* or they are *binary* to begin with.

4.4.1 Label Sequence Learning

Problem description In label sequence learning problems, the label set $\mathcal{Y}$ is a fixed alphabet containing the labels $l_1, l_2, \cdots, l_k$. The training set consists of $(\mathbf{x}_i, \mathbf{y}_i)_{i=1}^n$ where $\mathbf{y}_i$ is a sequence from the alphabet $\mathcal{Y}$. That is, $\mathbf{y}_i = (y_i^1, y_i^2, \cdots, y_i^{s_i})$, where, the length s_i of the sequence i need not be constant across all examples.

Joint feature map and kernel Considering the length of $\mathbf{y}$ to be T, the joint feature map for this problem is

$$\psi(\mathbf{x}, \mathbf{y}) = \begin{pmatrix} \sum_{t=1}^{T} \phi(\mathbf{x}^t) \otimes \lambda(y^t) \\ \sum_{t=1}^{T-1} \lambda(y^t) \otimes \lambda(y^{t+1}) \end{pmatrix},$$

where $\lambda(y^t) = \left[\delta(l_1, y^t), \delta(l_2, y^t) \cdots \delta(l_k, y^t)\right]^\top$. This feature map mimics the hidden Markov model (HMM) by considering the adjacent labels. These labels are equivalent to the hidden states in the HMM and the features are equivalent to HMM outputs. The kernel between two joint feature maps thus becomes

$$\psi(\mathbf{x}, \mathbf{y})^\top \psi(\bar{\mathbf{x}}, \bar{\mathbf{y}}) = \sum_{s=1}^{S} \sum_{t=1}^{T} \delta(y^t, \bar{y}^s) K(\mathbf{x}^t, \bar{\mathbf{x}}^s) + \sum_{s=1}^{S-1} \sum_{t=1}^{T-1} \delta(y^t, \bar{y}^s) \delta(y^{t+1}, \bar{y}^{s+1}),$$

where S is the length of the sequence $\bar{\mathbf{y}}$. These definitions of the joint feature map and the kernel between the joint feature maps are used in the experiments below.

	NER	POS
CRF	5.13 ± 0.28	11.34 ± 0.64
StructSVM	5.09 ± 0.32	11.14 ± 0.60
StructRMM	**5.05 ± 0.28**	**10.42 ± 0.47**
p-value	0.07	0.00

Table 4.1: Average percentage error rates (mean ± std. deviation) and the p-values on the two label sequence learning tasks. Improvements of StructRMM over StructSVM is of the same (or higher) order as that of StructSVM over CRF. A small p-value indicates statistical significance (for StructRMM over StructSVM).

Datasets and Evaluation To evaluate the sequence learning method, the first experiment involves the named entity recognition (NER) problem. A popular dataset for this task is from the Spanish news wire articles provided to the special NER session of the CoNLL 2002 conference. The words in these sentences are tagged with one of nine possible labels and the task is to predict the labels for each test sentence. In a second experiment, the parts-of-speech tagging task was considered using the sentences from the PennTree bank dataset. In this case, each word is labeled with one of the forty five possible parts-of-speech and the task is to accurately predict these tags for unseen test examples.

In both these experiments, the features used and the experimental setup are similar. The features describing words are all binary and follow the same protocols as previous work [Altun *et al.*, 2003b]. These binary features describe the context in which a word appears. For example, the features indicate whether the first letter of a word is capitalized, whether the word ends with a punctuation, whether the previous word has numbers in it and so on. The features were first generated for all the words independently and subsequently, a window of size three for each word was considered to incorporate information from adjacent words. The combined features in this window were aggregated for each word to form a window of features.

Random draws of size 240 from the corpus served as training sets and validation and test subsets of size 1000 were also drawn from the corpus. As in previous work, the polynomial

kernel of degree two was used in. The StructSVM was first trained using the cutting plane algorithm. After training, the maximum value of $|\mathbf{w}^\top \boldsymbol{\delta\psi}_i(\mathbf{y})|$ over all $\mathbf{y} \in \mathcal{Y}$, for any i, was noted. Subsequently, to evaluate the StructRMM, the B values were changed so that the maximum (minimum) allowed value of $\mathbf{w}^\top \boldsymbol{\delta\psi}_i(\mathbf{y})$ ($-\mathbf{w}^\top \boldsymbol{\delta\psi}_i(\mathbf{y})$) was within a fraction of the maximum found by the StructSVM. After training, the error rates were obtained on both validation and test sets for each setting of the parameters B and C. The setting which gave the lowest error on the validation set was used to determine the error rate on the test set (and vice versa). The experiment was repeated ten times over different random draws of the sets and the results were then averaged to obtain a final error rate for the StructSVM and the StructRMM.

Results Table 4.1 shows the results on both the tasks for CRF, StructSVM and StructRMM. The improvement of StructRMM over StructSVM on the named entity recognition task is nearly the same as that of StructSVM over CRF. However, in the case of parts-of-speech tagging experiment, the improvement of StructRMM over StructSVM is much higher than that of StructSVM over CRF. Statistical significance tests were conducted with the results from different folds for StructSVM and StructRMM. Small p-values obtained from a paired t-test indicate that these improvements are statistically significant.

4.4.2 Multi-class classification

Problem description For the multiclass-classification, the domain $\mathcal{Y}$ is a set with k possible values, for example $\{1, 2, 3, \cdots k\}$.

Joint feature map and kernel Given $(\mathbf{x}, \mathbf{y})$ where $\mathbf{y}_i$ is one of the k possible classes, the joint feature map for this problem is defined as the following:

$$\psi(\mathbf{x}, \mathbf{y}) := [\delta(1, \mathbf{y})\mathbf{x}^\top \delta(2, \mathbf{y})\mathbf{x}^\top \ \cdots \delta(k, \mathbf{y})\mathbf{x}^\top]^\top.$$

The joint feature map merely makes k "copies" of the features $\mathbf{x}$ but retains only one which is at the position of the position of the label $\mathbf{y}$. With this definition of the joint feature-map,

	Poly 1	Poly 2	Poly 3	Poly 4
StructSVM	3.78 ± 0.54	2.11 ± 0.43	1.73 ± 0.37	1.55 ± 0.45
StructRMM	3.85 ± 0.62	**1.46 ± 0.34**	**1.24 ± 0.43**	**1.18 ± 0.43**
p-value	0.55	0.00	0.00	0.00

Table 4.2: Average percentage error rates (mean ± std. deviation) and the p-values on the OPTDIGIT multi-class problem. Negligible p-values indicate statistical significance. Average improvement of StructRMM over StructSVM is about 20% and the improvement is about 28% excluding the linear kernel.

the kernel between $(\mathbf{x}, \mathbf{y})$ and $(\bar{\mathbf{x}}, \bar{\mathbf{y}})$ becomes:

$$\psi(\mathbf{x}, \mathbf{y})^\top \psi(\bar{\mathbf{x}}, \bar{\mathbf{y}}) = \sum_{i=1}^{k} \delta(i, \mathbf{y}) \delta(i, \bar{\mathbf{y}}) K(\mathbf{x}, \bar{\mathbf{x}}).$$

In the multi-class problems, the arg max (arg min) in the cutting plane algorithm 4.1 can be found by simple enumeration due to the small size of the output space.

Dataset and Evaluation In the multi-class experiment, the Optical Digits [Asuncion and Newman, 2007] was considered which contains 3823 digits represented via 64 scalar features that measure pixel intensities. The dataset was first normalized so that each feature was in the range $[0, 1]$.

Half of the examples in the database were randomly drawn to form a training set and the remaining examples were divided, randomly, into two sets of equal size to form validation and test sets. The StructSVM was first trained by exploring various C values. For the StructSVM various values were explored for both C and B. The setting of the parameters B and C which achieved minimum error on the validation set was used to pick an error rate for StructSVM and StructRMM from the test set (and vice versa). The entire experiment was repeated for ten different draws of the training, test and validation sets. Furthermore, the experiments were performed for polynomial kernel with degrees one, two, three and four.

Results The results are presented in Table 4.2. With the linear kernel, StructRMM performed at the same level as StructSVM. But with the polynomial kernel degrees two, three and four the performance of StructRMM was significantly better than that of StructSVM. As before, statistical significance tests were conducted using the paired t-test. The tests suggest that there is no significant difference between StructSVM and StructRMM in the case of linear kernel. However, for the higher degree kernels, the p-values were negligible indicating statistical significance. In fact, the average improvement of StructRMM over StructSVM with these three kernels is about 28%.

4.5 Summary

A shortcoming of the large absolute margin method and its sensitivity to scaling was pointed out for structured prediction problems. Subsequently, a formulation was proposed to maximize the margin relative to the spread of the data. An efficient cutting plane algorithm, which can be solved in nearly same time as the StructSVM, was then proposed. Experimental results on label sequence learning problems show that the improvement of StructRMM over StructSVM is of the same order as that of StructSVM over CRF. Dramatic improvements were obtained on multi-class classification problems as well. Additionally, the improvement in accuracy brought forth by the relative margin method does not carry significant increases in computational complexity.

Chapter 5

Laplacian spectrum learning

So far, the discussion was limited to fully supervised problems where all the training examples were labeled. In this chapter, a setting where a set of labeled examples is accompanied by a set of unlabeled examples is considered. This is known as the transductive setup; here the aim is to label the unlabeled examples. However, both labeled and unlabeled examples could be exploited during learning. In this chapter, the unlabeled examples will be utilized in estimating a kernel matrix on both the labeled and the unlabeled examples.

One particularly successful approach for estimating such a kernel matrix is by transforming the spectrum of the graph Laplacian [Smola and Kondor, 2003]. Graph Laplacians have many interesting properties. For example, eigenvectors of a graph Laplacian corresponding to smaller eingenvalues are "smoother" over the graph compared to eigenvectors corresponding to higher eigenvalues. Thus, a natural way to build a kernel preserving the smoothness of the graph is by monotonically inverting the spectrum of the graph Laplacian. In fact, the diffusion kernel [Kondor and Lafferty, 2002] and the Gaussian field kernel [Zhu *et al.*, 2003] are based on such an approach and explore smooth variations of the Laplacian via specific parametric forms.

Kernel target alignment (KTA for short) [Cristianini *et al.*, 2001] is a criterion for evaluating a kernel based on the labels. It was initially proposed as a method to choose a kernel from a family of candidates such that the Frobenius norm of the difference between a label matrix and the kernel matrix is minimized. This technique estimates a kernel independently of the final learning algorithm for classification. Recently, such a method

was proposed to transform the spectrum of a graph Laplacian [Zhu *et al.*, 2004] to select from a general family of candidate kernels. Instead of relying on parametric methods for exploring a family of kernels (such as the scalar parameter in a diffusion or Gaussian field kernel), [Zhu *et al.*, 2004] suggest a more general approach which yields a kernel matrix non-parametrically that aligns well with an ideal kernel (obtained from the labeled examples). In this chapter, spectrum of the graph Laplacian are transformed to optimize the large (relative) margin criterion.

The motivation for large (relative) margin spectrum transformation is straightforward. In kernel target alignment, a simpler surrogate criterion is first optimized to obtain a kernel by transforming the graph Laplacian. Then, the kernel obtained is fed to a classifier such as an SVM. This is a two-step process with a different objective function in each step. It is more natural to transform the Laplacian spectrum jointly with the classification criterion in the first place rather than using a surrogate criterion to learn the kernel.

5.1 Setup and notation

In this chapter it is assumed that a set of labeled examples $(\mathbf{x}_i, y_i)_{i=1}^{l}$ and an unlabeled set $(\mathbf{x}_i)_{i=l+1}^{n}$ are given such that $\mathbf{x}_i \in \mathbb{R}^m$ and $y_i \in \{\pm 1\}$. The vector $\mathbf{y} \in \mathbb{R}^l$ is such that its i^{th} entry is y_i. Moreover, $\mathbf{Y} \in \mathbb{R}^{l \times l}$ is a diagonal matrix such that $\mathbf{Y}_{ii} = y_i$. The primary aim is to obtain predictions on the unlabeled examples. However, the unlabeled examples can also be utilized in the learning process.

Assume that a graph is denoted by its adjacency matrix $\mathbf{W} \in \mathbb{R}^{n \times n}$ where the weight $\mathbf{W}_{ij}$ denotes the edge weight between nodes i and j (corresponding to the examples $\mathbf{x}_i$ and $\mathbf{x}_j$). Define the graph Laplacian as $\mathbf{L} = \mathbf{D} - \mathbf{W}$ where $\mathbf{D}$ denotes a diagonal matrix whose i^{th} entry is given by the sum of the i^{th} row of $\mathbf{W}$. The eigen decomposition of $\mathbf{L}$ is given by $\mathbf{L} = \sum_{i=1}^{n} \theta_i \boldsymbol{\phi}_i \boldsymbol{\phi}_i^\top$; it is assumed that the eigenvalues are already arranged such that $\theta_i \leq \theta_{i+1}$ for all i. Further, $\mathbf{V} \in \mathbb{R}^{n \times q}$ is a matrix whose i^{th} column is the $(i+1)^{\text{th}}$ eigenvector (corresponding to the $(i+1)^{\text{th}}$ smallest eigenvalue) of $\mathbf{L}$. Note that the first eigenvector (corresponding to the smallest eigenvalue) has been deliberately left out from this definition. Further, $\mathbf{U} \in \mathbb{R}^{n \times q}$ is defined to be the matrix whose i^{th} column is the

i^{th} eigenvector. $\mathbf{v}_i$ ($\mathbf{u}_i$) denotes the i^{th} column of $\mathbf{V}^\top$ ($\mathbf{U}^\top$). For any eigenvector (such as $\boldsymbol{\phi}, \mathbf{u}$ or $\mathbf{v}$), corresponding notation with a bar on top ($\bar{\phi}, \bar{\mathbf{u}}$ or $\bar{\mathbf{v}}$) denotes the (first l) elements of the vector corresponding to the labeled examples. This notation is overloaded for matrices as well; thus $\bar{\mathbf{V}} \in \mathbb{R}^{l \times q}$ denotes[1] the first l rows of $\mathbf{V}$. $\boldsymbol{\Delta}$ is assumed to be a $q \times q$ diagonal matrix, whose diagonal elements are given by $\boldsymbol{\Delta}_{ii} = \delta_i$. Finally $\mathbf{0}$ and $\mathbf{1}$ denote vectors of all zeros and all ones; their dimensionality can be inferred from the context.

The rest of this chapter is organized as follows. Various approaches that make use of the graph Laplacian in learning are introduced in Section 5.2. Motivation for Laplacian spectrum learning is provided in Section 5.3. The main contribution is in Section 5.4 where the proposed optimization problems are derived. Experiments on real world datasets are presented in Section 5.5. This chapter ends with a summary in Section 5.6. This chapter is based on the work that will be published in [Shivaswamy and Jebara, 2010a].

5.2 Learning from the graph Laplacian

Graph Laplacian has been particularly popular in transductive learning. Several approaches have been proposed in the literature that exploit the properties of a Laplacian. Some popular approaches that exploit the graph Laplacian and that are relevant in the context of this chapter are discussed below.

Spectral Graph Transducer The spectral graph transducer [Joachims, 2003] is a trasductive method based on the relaxation of a graph-cut point of view. In a nutshell, it obtains all the predictions (on labeled as well as unlabeled examples) by solving for $\mathbf{h} \in \mathbb{R}^n$ via the following problem:

$$\begin{aligned} \min_{\mathbf{h} \in \mathbb{R}^n} \quad & \frac{1}{2}\mathbf{h}^\top \mathbf{V}\mathbf{Q}\mathbf{V}^\top \mathbf{h} + C(\mathbf{h} - \boldsymbol{\tau})^\top \mathbf{P}(\mathbf{h} - \boldsymbol{\tau}) \\ \text{s.t. } & \mathbf{h}^\top \mathbf{1} = 0, \quad \mathbf{h}^\top \mathbf{h} = n. \end{aligned} \tag{5.1}$$

where $\mathbf{P}$ is a diagonal matrix[2] with $\mathbf{P}_{ii} = \frac{1}{l_+}$ ($\frac{1}{l_-}$) if the i^{th} example is positive (negative); $\mathbf{P}_{ii} = 0$ for unlabeled examples (i.e., for $l+1 \leq i \leq n$). Further, $\mathbf{Q}$ is also a diagonal matrix

[1] It is clarified that $\bar{\mathbf{V}}^\top$ denotes the transpose of $\bar{\mathbf{V}}$.

[2] $l_+(l_-)$ is the number of positive (negative) labeled examples.

whose diagonal values are set by $\mathbf{Q}_{ii} = i^2$ in the case of spectral graph transducer. $\boldsymbol{\tau}$ is a vector in which the values corresponding to the positive (negative) examples are set to $\sqrt{\frac{l_-}{l_+}}$ ($\sqrt{\frac{l_+}{l_-}}$). It is thus interesting to note that the spectral graph transducer is actually monotonically transforming the spectrum of a Laplacian (via the transformation $\mathbf{Q}_{ii} = i^2$).

Non-parametric transformations via kernel target alignment (KTA) A particularly successful approach to learning a kernel by transforming the spectrum of a Laplacian in a non-parametric way was proposed in [Zhu *et al.*, 2004]. The empirical alignment between two kernel matrices $\mathbf{K}_1$ and $\mathbf{K}_2$ is defined as [Cristianini *et al.*, 2001]:

$$\hat{A}(\mathbf{K}_1, \mathbf{K}_2) := \frac{\langle \mathbf{K}_1, \mathbf{K}_2 \rangle_F}{\sqrt{\langle \mathbf{K}_1, \mathbf{K}_1 \rangle_F \langle \mathbf{K}_2, \mathbf{K}_2 \rangle_F}}.$$

When the target $\mathbf{y}$ (the vector formed by concatenating y_i's) is known, the ideal kernel matrix is $\mathbf{y}\mathbf{y}^\top$ and a kernel matrix $\mathbf{K}$ can be learned by maximizing the alignment $\hat{A}(\bar{\mathbf{K}}, \mathbf{y}\mathbf{y}^\top)$. The kernel target alignment approach to learning a kernel [Zhu *et al.*, 2004] is via the following formulation:

$$\begin{aligned} \max_{\boldsymbol{\Delta}} \ & \hat{A}(\bar{\mathbf{U}}\boldsymbol{\Delta}\bar{\mathbf{U}}^\top, \mathbf{y}\mathbf{y}^\top) \\ \text{s.t. } & \operatorname{trace}(\mathbf{U}\boldsymbol{\Delta}\mathbf{U}^\top) = 1, \\ & \delta_i \geq \delta_{i+1} \qquad \forall 2 \leq i \leq q-1, \\ & \delta_q \geq 0, \quad \delta_1 \geq 0. \end{aligned} \tag{5.2}$$

The above optimization problem transforms the spectrum of the given graph Laplacian $\mathbf{L}$ while maximizing the alignment score with the label part of the kernel matrix ($\bar{\mathbf{U}}\boldsymbol{\Delta}\bar{\mathbf{U}}^\top$) with the true labels. The trace constraint on the overall kernel matrix ($\mathbf{U}\boldsymbol{\Delta}\mathbf{U}^\top$) is to prevent arbitrary scaling. The above formulation can be posed as a quadratically constrained quadratic program (QCQP) that can be solved efficiently [Zhu *et al.*, 2004]. The ordering on the δ's is in reverse order as that of the eigenvalues of $\mathbf{L}$ which amounts to monotonically inverting the spectrum of the graph Laplacian $\mathbf{L}$. Only the first q eigenvectors are considered in the formulation above due to computational considerations.

Note that the ordering constraint (on δ's) in (5.2) excludes δ_1. This is because the eigenvector $\boldsymbol{\phi}_1$ is made up of a constant element. Thus, it merely amounts to adding a

constant element to all the elements of the kernel matrix. Thus, the weight on this (δ_1) is allowed to vary freely. Finally, since $\boldsymbol{\phi}$'s are the eigenvectors of $\mathbf{L}$, the trace constraint on $\mathbf{U}\boldsymbol{\Delta}\mathbf{U}^\top$ merely corresponds to the constraint $\sum_{i=1}^{q} \delta_i = 1$ since $\mathbf{U}^\top\mathbf{U} = \mathbf{I}$.

Parametric transformations A number of methods have been proposed to obtain a kernel from the graph Laplacian. These methods essentially take the Laplacian from labeled and unlabeled data and transform the spectrum with a particular form. A natural way to build a kernel is by $\mathbf{K} = \sum_{i=1}^{n} r(\theta_i)\boldsymbol{\phi}_i\boldsymbol{\phi}_i^\top$ where $r(\cdot)$ is a monotonically decreasing function. Thus, an eigenvector with a small eigenvalue will have a large weight in the kernel matrix. For example, the diffusion kernel [Kondor and Lafferty, 2002] is obtained by the transformation $r(\theta) = \exp(-\theta/\sigma^2)$, Gaussian field kernel [Zhu *et al.*, 2003] uses the transformation $r(\theta) = \frac{1}{\sigma^2+\theta}$. In fact, kernel PCA [Schölkopf *et al.*, 1998] also performs a similar operation. In kPCA, the top k eigenvectors of a kernel matrix are retained. From an equivalence that exists between the kernel matrix and the graph Laplacian (shown in the next section), it can be concluded that kernel PCA features also fall under the same family of monotonic transformations. While these are very interesting transformations, since [Zhu *et al.*, 2004] showed that KTA based approaches are empirically superior to parametric transformations, these parametric approaches are not discussed further in this chapter.

5.3 Why learn the Laplacian spectrum?

An optimization problem closely related to the spectral graph transducer (5.1) is considered in this section. The main difference is in the choice of the loss function. Further, there are no additional constraints on $\mathbf{h}$ either in the following optimization problem:[3]

$$\min_{\mathbf{h}\in\mathbb{R}^n} \frac{1}{2}\mathbf{h}^\top\mathbf{V}\boldsymbol{\Delta}^{-1}\mathbf{V}^\top\mathbf{h} + C\sum_{i=1}^{l}\max(0, 1 - y_i\mathbf{h}_i). \tag{5.3}$$

The above optimization problem is essentially learning predictions on all the examples by minimizing the hinge loss with a regularization based on the eigenspace of the graph Laplacian. The choice of the above formulation is due to its relation to the large margin learning framework given by the following theorem.

[3]Here $\boldsymbol{\Delta}$ is assumed to be invertible. For instance, if $\boldsymbol{\Delta}_{ii}^{-1} = \theta_{i+1}$, $\boldsymbol{\Delta}$ is invertible for a connected graph.

Theorem 22 *The optimization problem* (5.3) *is equivalent to*

$$\min_{\mathbf{w},b} \ \frac{1}{2}\mathbf{w}^\top\mathbf{w} + C\sum_{i=1}^{l}\max(0, 1 - y_i(\mathbf{w}^\top\boldsymbol{\Delta}^{\frac{1}{2}}\mathbf{v}_i + b)). \tag{5.4}$$

Proof The predictions on all the examples (without the bias term) for the optimization problem (5.4) are given by $\mathbf{f} = \mathbf{V}\boldsymbol{\Delta}^{\frac{1}{2}}\mathbf{w}$. Therefore $\boldsymbol{\Delta}^{-\frac{1}{2}}\mathbf{V}^\top\mathbf{f} = \boldsymbol{\Delta}^{-\frac{1}{2}}\mathbf{V}^\top\mathbf{V}\boldsymbol{\Delta}^{\frac{1}{2}}\mathbf{w} = \mathbf{w}$ since $\mathbf{V}^\top\mathbf{V} = \mathbf{I}$. Substituting this expression for $\mathbf{w}$ in (5.4), the optimization problem becomes,

$$\min_{\mathbf{f},b} \ \frac{1}{2}\mathbf{f}^\top\mathbf{V}\boldsymbol{\Delta}^{-1}\mathbf{V}^\top\mathbf{f} + C\sum_{i=1}^{l}\max(0, 1 - y_i(\mathbf{f}_i + b)).$$

Let $\mathbf{h} = \mathbf{f} + b\mathbf{1}$ and consider the first term in the objective above,

$$\begin{aligned}
&(\mathbf{h} - b\mathbf{1})^\top\mathbf{V}\boldsymbol{\Delta}^{-1}\mathbf{V}^\top(\mathbf{h} - b\mathbf{1}) \\
=&\mathbf{h}^\top\mathbf{V}\boldsymbol{\Delta}^{-1}\mathbf{V}^\top\mathbf{h} + 2\mathbf{h}^\top\mathbf{V}\boldsymbol{\Delta}^{-1}\mathbf{V}^\top\mathbf{1} + \mathbf{1}^\top\mathbf{V}\boldsymbol{\Delta}^{-1}\mathbf{V}^\top\mathbf{1} \\
=&\mathbf{h}^\top\mathbf{V}\boldsymbol{\Delta}^{-1}\mathbf{V}^\top\mathbf{h},
\end{aligned}$$

where the fact that $\mathbf{V}^\top\mathbf{1} = \mathbf{0}$ has been exploited. This is because $\mathbf{1}$ is always an eigenvector of $\mathbf{L}$ and other eigenvectors are orthogonal to it. Thus, the optimization problem (5.3) follows. ■

By the above theorem, thus, learning predictions with a Laplacian regularization in (5.3) is equivalent to learning in a large margin setting (5.4). It is easy to see that the implicit kernel for the learning algorithm (5.4) (over both labeled and unlabeled examples) is given by $\mathbf{V}\boldsymbol{\Delta}\mathbf{V}^\top$. Thus, obtaining predictions on all the examples with $\mathbf{V}\boldsymbol{\Delta}^{-1}\mathbf{V}^\top$ as the regularizer on predictions in (5.3) is equivalent to large margin learning with the kernel obtained by inverting the spectrum $\boldsymbol{\Delta}$. However, it is not at all clear as to why an inverted spectrum is the right choice for obtaining a kernel from the Laplacian spectrum. Parametric methods discussed in the previous section construct this kernel by particular parametric forms. Kernel target alignment based approach constructs this kernel by maximizing the alignment with the labels while maintaining an order on the spectrum. Instead, formulations are proposed in this chapter to explore a family of transformations but the algorithm is allowed to choose the right transformation which is suited to a large (relative) margin criterion. Although $\boldsymbol{\phi}_1$ is excluded in the definition of $\mathbf{V}$ in the above derivation, typically it is

included in the optimization (thus $\mathbf{U}$ will be used). However, the weight on this eigenvector is allowed to vary freely as in the kernel target alignment approach. Moreover, experiments show that the algorithms typically choose a negligible weight on this eigenvector.

5.3.1 RMM on Laplacian eigenmaps

Based on the motivation from earlier sections, consider the problem of jointly learning a classifier and weights on various eigenvectors in the RMM setup. The family of weights is restricted to be the same as that in (5.2):

$$\begin{aligned}
\min_{\mathbf{w},b,\xi,\boldsymbol{\Delta}} \quad & \frac{1}{2}\mathbf{w}^\top\mathbf{w} + C\sum_{i=1}^{l}\xi_i && && (5.5)\\
\text{s.t. } & y_i(\mathbf{w}^\top\boldsymbol{\Delta}^{\frac{1}{2}}\mathbf{u}_i + b) \geq 1-\xi_i,\ \xi_i \geq 0 && \forall 1 \leq i \leq l\\
& -(\mathbf{w}^\top\boldsymbol{\Delta}^{\frac{1}{2}}\mathbf{u}_i + b) \leq B && \forall 1 \leq i \leq l\\
& +(\mathbf{w}^\top\boldsymbol{\Delta}^{\frac{1}{2}}\mathbf{u}_i + b) \leq B && \forall 1 \leq i \leq l\\
& \delta_i \geq \delta_{i+1} && \forall 2 \leq i \leq q-1\\
& \delta_1 \geq 0, \quad \delta_q \geq 0\\
& \operatorname{trace}(\mathbf{U}\boldsymbol{\Delta}\mathbf{U}^\top) = 1.
\end{aligned}$$

The aim now is to express the above optimization in a standard form. Towards this end, the dual of the above optimization problem is derived with respect to the variables $\mathbf{w}$, b and ξ while still retaining $\boldsymbol{\Delta}$ in the primal form. This results in the following optimization problem:

$$\begin{aligned}
\min_{\boldsymbol{\Delta}} \max_{\boldsymbol{\alpha},\boldsymbol{\beta},\boldsymbol{\eta}} \quad & -\frac{1}{2}\boldsymbol{\gamma}^\top\bar{\mathbf{U}}\boldsymbol{\Delta}\bar{\mathbf{U}}^\top\boldsymbol{\gamma} + \boldsymbol{\alpha}^\top\mathbf{1} - B\left(\boldsymbol{\beta}^\top\mathbf{1} + \boldsymbol{\eta}^\top\mathbf{1}\right) && && (5.6)\\
\text{s.t. } & \boldsymbol{\alpha}^\top\mathbf{y} - \boldsymbol{\beta}^\top\mathbf{1} + \boldsymbol{\eta}^\top\mathbf{1} = 0\\
& \mathbf{0} \leq \boldsymbol{\alpha} \leq C\mathbf{1}, \quad \boldsymbol{\beta} \geq \mathbf{0}, \quad \boldsymbol{\eta} \geq \mathbf{0}\\
& \delta_i \geq \delta_{i+1} && \forall 2 \leq i \leq q-1\\
& \delta_1 \geq 0,\ \delta_q \geq 0\\
& \sum_{i=1}^{q}\delta_i = 1.
\end{aligned}$$

In the above optimization problem, the fact that trace($\mathbf{U}\boldsymbol{\Delta}\mathbf{U}^\top$) = $\sum_{i=1}^{q}\delta_i$ has been exploited. Further, for brevity, $\boldsymbol{\gamma}$ denotes $\mathbf{Y}\boldsymbol{\alpha} - \boldsymbol{\beta} + \boldsymbol{\eta}$. The above optimization problem without the ordering constraints (*i.e.,* $\delta_i \geq \delta_{i+1}$) is the well known multiple kernel learning problem [Bach *et al.*, 2004][4] (using RMM criterion instead of the standard SVM). In this special case, a straightforward derivation–following the approach of [Bach *et al.*, 2004]–results in the corresponding multiple kernel learning optimization. Even though the optimization problem (5.6) without the ordering on δ's is a more general problem, it may not be very helpful to obtain predictions that are smooth over the entire graph. This is because, with a small number of labeled examples (*i.e.,* small l), it is unlikely that multiple kernel learning will recover an ordering unless enforced. In fact, this phenomenon can be observed in the experiments where multiple kernel learning fails to recover a meaningful order on the spectrum.

5.4 STORM and STOAM

The aim here is to start with the optimization problem (5.6) and to pose it as a standard optimization problem. The min and the max in (5.6) can be interchanged since the objective is concave in $\boldsymbol{\Delta}$ and convex in $\boldsymbol{\alpha}$, $\boldsymbol{\beta}$ and $\boldsymbol{\eta}$ and both are strictly feasible[5] [Boyd and Vandenberghe, 2003][6]. Thus, the optimization problem (5.6) can be equivalently written

[4] In this chapter comparisons are made only to convex combination multiple kernel learning algorithms.

[5] It is trivial to construct such $\boldsymbol{\alpha}$, $\boldsymbol{\beta}$, $\boldsymbol{\eta}$ and $\boldsymbol{\Delta}$ when not all the labels are the same.

[6] See the section on the Lagrange dual problem.

as:

$$\max_{\boldsymbol{\alpha},\boldsymbol{\beta},\boldsymbol{\eta}} \min_{\boldsymbol{\Delta}} \; -\frac{1}{2}\boldsymbol{\gamma}^\top \sum_{i=1}^{q} \delta_i \bar{\mathbf{u}}_i \bar{\mathbf{u}}_i^\top \boldsymbol{\gamma} + \boldsymbol{\alpha}^\top \mathbf{1} - B\left(\boldsymbol{\beta}^\top \mathbf{1} + \boldsymbol{\eta}^\top \mathbf{1}\right) \tag{5.7}$$

$$\begin{aligned}
&\text{s.t. } \boldsymbol{\alpha}^\top \mathbf{y} - \boldsymbol{\beta}^\top \mathbf{1} + \boldsymbol{\eta}^\top \mathbf{1} = 0 \\
&\quad \mathbf{0} \le \boldsymbol{\alpha} \le C\mathbf{1}, \;\; \boldsymbol{\beta} \ge \mathbf{0}, \;\; \boldsymbol{\eta} \ge \mathbf{0} \\
&\quad \delta_i \ge \delta_{i+1} \qquad\qquad \forall 2 \le i \le q-1 \\
&\quad \delta_1 \ge 0, \; \delta_q \ge 0 \\
&\quad \sum_{i=1}^{q} \delta_i = 1.
\end{aligned}$$

5.4.1 An unsuccessful attempt

An attempt is first made to directly obtain the optimization problem (5.7) in a standard form. However, it turns out that this approach does not work. Consider the inner optimization over $\boldsymbol{\Delta}$ in the above optimization problem (5.7):

$$\min_{\boldsymbol{\Delta}} \; -\frac{1}{2}\sum_{i=1}^{q} \delta_i \boldsymbol{\gamma}^\top \bar{\mathbf{u}}_i \bar{\mathbf{u}}_i^\top \boldsymbol{\gamma} \tag{5.8}$$

$$\begin{aligned}
&\text{s.t. } \delta_i \ge \delta_{i+1} \qquad\qquad \forall 2 \le i \le q-1 \\
&\quad \delta_1 \ge 0, \;\; \delta_q \ge 0 \\
&\quad \sum_{i=1}^{q} \delta_i = 1.
\end{aligned}$$

Lemma 23 *The dual of the above formulation is:*

$$\begin{aligned}
&\max_{\tau,\boldsymbol{\lambda}} \; -\tau \\
&\quad s.t. \; \frac{1}{2}\boldsymbol{\gamma}^\top \bar{\mathbf{u}}_i \bar{\mathbf{u}}_i^\top \boldsymbol{\gamma} = \lambda_{i-1} - \lambda_i + \tau, \; \lambda_i \ge 0 \qquad\qquad \forall 1 \le i \le q.
\end{aligned}$$

where $\lambda_0 = 0$ is a dummy variable.

Proof Start by writing the Lagrangian of the optimization problem:

$$\mathcal{L} = -\frac{1}{2}\sum_{i=1}^{q} \delta_i \boldsymbol{\gamma}^\top \bar{\mathbf{u}}_i \bar{\mathbf{u}}_i^\top \boldsymbol{\gamma} - \sum_{i=2}^{q-1} \lambda_i(\delta_i - \delta_{i+1}) - \lambda_q \delta_q - \lambda_1 \delta_1 + \tau(\sum_{i=1}^{q} \delta_i - 1),$$

where $\lambda_i \geq 0$ and τ are the Lagrange multipliers. The dual follows from differentiating $\mathcal{L}$ with respect to δ_i and equating the resulting expression to zero. ■

While the above dual is independent of δ's, the constraints $\frac{1}{2}\boldsymbol{\gamma}^\top \bar{\mathbf{u}}_i \bar{\mathbf{u}}_i^\top \boldsymbol{\gamma} = \lambda_{i-1} - \lambda_i + \tau$ involve a quadratic term in an equality. It is not possible to simply leave out λ_i to make this constraint an inequality, since the same λ_i occurs in two equations. This is non-convex in $\boldsymbol{\gamma}$ and is problematic since, after all, a jointly convex optimization in $\boldsymbol{\gamma}$ and other variables is desirable. Thus, a reformulation is necessary to pose relative margin kernel learning as a jointly convex optimization problem.

5.4.2 A refined approach

To circumvent the problem mentioned above, an alternative approach is considered now. The ordering on the eigenvalues is still maintained, however, they are required to be at least ϵ apart. Here $\epsilon > 0$ is a (fixed) constant. It can be tiny (for example, 10^{-6} in the experiments) but has to be positive. Further, the smallest eigenvalue is required to be at least ϵ in the following problem:

$$\begin{aligned} \min_{\boldsymbol{\Delta}} \; & -\frac{1}{2}\sum_{i=1}^{q} \delta_i \boldsymbol{\gamma}^\top \bar{\mathbf{u}}_i \bar{\mathbf{u}}_i^\top \boldsymbol{\gamma} && (5.9) \\ \text{s.t. } & \delta_i - \delta_{i+1} \geq \epsilon && \forall 2 \leq i \leq q-1 \\ & \delta_1 \geq \epsilon, \quad \delta_q \geq \epsilon \\ & \sum_{i=1}^{q} \delta_i = 1. \end{aligned}$$

Since ϵ is a fixed constant, no new parameters are introduced in posing the above problem in a standard form. The following theorem shows that certain change of variables can be done in the above optimization problem so that its dual is in a particularly convenient form.

Theorem 24 *The dual of the optimization problem* (5.9) *is:*

$$\begin{aligned} \max_{\boldsymbol{\lambda}\geq 0,\tau} \quad & -\tau + \epsilon \sum_{i=1}^{q} \lambda_i \\ s.t. \ & \frac{1}{2}\boldsymbol{\gamma}^\top \sum_{j=2}^{i} \bar{\mathbf{u}}_j \bar{\mathbf{u}}_j^\top \boldsymbol{\gamma} = \tau(i-1) - \lambda_i \qquad \forall 2 \leq i \leq q \\ & \frac{1}{2}\boldsymbol{\gamma}^\top \bar{\mathbf{u}}_1 \bar{\mathbf{u}}_1^\top \boldsymbol{\gamma} = \tau - \lambda_1. \end{aligned} \tag{5.10}$$

Proof Start with the following change of variables:

$$\kappa_i := \begin{cases} \delta_1 & \text{for } i = 1, \\ \delta_i - \delta_{i+1} & \text{for } 2 \leq i \leq q-1, \\ \delta_q & \text{for } i = q. \end{cases}$$

This gives:

$$\delta_i = \begin{cases} \kappa_1 & \text{for } i = 1, \\ \sum_{j=i}^{q} \kappa_j & \text{for } 2 \leq i \leq q. \end{cases} \tag{5.11}$$

Thus, (5.9) can be stated as,

$$\begin{aligned} \min_{\kappa} \ & -\frac{1}{2} \sum_{i=2}^{q} \sum_{j=i}^{q} \kappa_j \boldsymbol{\gamma}^\top \bar{\mathbf{u}}_i \bar{\mathbf{u}}_i^\top \boldsymbol{\gamma} + \kappa_1 \boldsymbol{\gamma}^\top \bar{\mathbf{u}}_1 \bar{\mathbf{u}}_1^\top \boldsymbol{\gamma} \\ \text{s.t. } & \kappa_i \geq \epsilon \quad \forall 1 \leq i \leq q, \\ & \sum_{i=2}^{q} \sum_{j=i}^{q} \kappa_j + \kappa_1 = 1. \end{aligned} \tag{5.12}$$

Consider simplifying the following term within the above formulation:

$$\sum_{i=2}^{q} \sum_{j=i}^{q} \kappa_j \boldsymbol{\gamma}^\top \bar{\mathbf{u}}_i \bar{\mathbf{u}}_i^\top \boldsymbol{\gamma} = \sum_{i=2}^{q} \kappa_i \sum_{j=2}^{i} \boldsymbol{\gamma}^\top \bar{\mathbf{u}}_j \bar{\mathbf{u}}_j^\top \boldsymbol{\gamma},$$

and

$$\sum_{i=2}^{q} \sum_{j=i}^{q} \kappa_j = \sum_{i=2}^{q} (i-1)\kappa_i.$$

It is now straightforward to write the Lagrangian to obtain the dual. ■

Even though the above optimization appears to have non-convexity problems mentioned after Lemma 23, these can be avoided. This is facilitated by the following nice property.

Lemma 25 *For $\epsilon > 0$, all the inequality constraints are active at optimum in the following optimization problem:*

$$\begin{aligned} \max_{\boldsymbol{\lambda}\geq\mathbf{0},\tau} \quad & -\tau + \epsilon \sum_{i=1}^{q} \lambda_i \\ s.t. \ & \frac{1}{2}\boldsymbol{\gamma}^\top \sum_{j=2}^{i} \bar{\mathbf{u}}_j \bar{\mathbf{u}}_j^\top \boldsymbol{\gamma} \leq \tau(i-1) - \lambda_i \qquad \forall 2 \leq i \leq q \\ & \frac{1}{2}\boldsymbol{\gamma}^\top \bar{\mathbf{u}}_1 \bar{\mathbf{u}}_1^\top \boldsymbol{\gamma} \leq \tau - \lambda_1. \end{aligned} \tag{5.13}$$

Proof Assume that $\boldsymbol{\lambda}^*$ is the optimum for the above problem and constraint i (corresponding to λ_i) is not active, then clearly, the objective can be further maximized by increasing λ_i^*. This contradicts the fact that $\boldsymbol{\lambda}^*$ is the optimum. ■

In fact, it is not hard to show that the Lagrange multipliers of the constraints in problem (5.13) are equal to the κ_i's. Thus, replacing the inner optimization over δ's in (5.7), by (5.13), the following optimization problem is obtained:

$$\begin{aligned} \max_{\boldsymbol{\alpha},\boldsymbol{\beta},\boldsymbol{\eta},\boldsymbol{\lambda},\tau} \quad & \boldsymbol{\alpha}^\top \mathbf{1} - \tau + \epsilon \sum_{i=1}^{q} \lambda_i - B\left(\boldsymbol{\beta}^\top \mathbf{1} + \boldsymbol{\eta}^\top \mathbf{1}\right) \\ \text{s.t. } & \frac{1}{2}(\mathbf{Y}\boldsymbol{\alpha} - \boldsymbol{\beta} + \boldsymbol{\eta})^\top \sum_{j=2}^{i} \bar{\mathbf{u}}_j \bar{\mathbf{u}}_j^\top (\mathbf{Y}\boldsymbol{\alpha} - \boldsymbol{\beta} + \boldsymbol{\eta}) \leq (i-1)\tau - \lambda_i \quad \forall 2 \leq i \leq q \\ & \frac{1}{2}(\mathbf{Y}\boldsymbol{\alpha} - \boldsymbol{\beta} + \boldsymbol{\eta})^\top \bar{\mathbf{u}}_1 \bar{\mathbf{u}}_1^\top (\mathbf{Y}\boldsymbol{\alpha} - \boldsymbol{\beta} + \boldsymbol{\eta}) \leq \tau - \lambda_1 \\ & \boldsymbol{\alpha}^\top \mathbf{y} - \boldsymbol{\beta}^\top \mathbf{1} + \boldsymbol{\eta}^\top \mathbf{1} = 0 \\ & \mathbf{0} \leq \boldsymbol{\alpha} \leq C\mathbf{1}, \quad \boldsymbol{\beta} \geq \mathbf{0}, \quad \boldsymbol{\eta} \geq \mathbf{0}, \quad \boldsymbol{\lambda} \geq \mathbf{0}. \end{aligned} \tag{5.14}$$

The above optimization problem will be referred to as STORM (Spectrum Transformations that Optimize the Relative Margin). It has a linear objective with quadratic constraints; this optimization problem falls under the family quadratically constrained quadratic optimization (QCQP). When $B = \infty$, the above optimization problem reduces to STOAM (Spectrum Transformations that Optimize the Absolute Margin) since there is no constraint on the spread of the projections (the same way an RMM mimics an SVM when $B = \infty$).

Obtaining δ values Interior point methods obtain both primal and dual solutions of an optimization problem simultaneously. Thus, it is possible to recover the weights on each eigenvector via the equation (5.11).

Computational complexity STORM is a standard QCQP with q quadratic constraints of dimensionality l. This can be solved in time $\mathcal{O}(ql^3)$ with an interior point solver [Boyd and Vandenberghe, 2003]. It is important to note that the number of labeled examples l is much smaller than the total number of examples (which is n). Moreover, q is typically a fixed constant. Thus the runtime of the proposed QCQP compares favorably with the $\mathcal{O}(n^3)$ time for the initial eigen decomposition of $\mathbf{L}$ which is required for all the methods described in this chapter.

5.5 Experiments

To study the empirical performance of STORM and STOAM with respect to previous work, experiments were performed on text and digit classification problems. Five binary classification problems were chosen from the 20-newsgroups text dataset (separating categories like baseball-hockey (b-h), pc-mac (p-m), religion-atheism (r-a), windows-xwindows (w-x), and politics.mideast-politics.misc (m-m)). Similarly, five different problems were considered from the MNIST dataset (0-9, 1-2, 3-8, 4-7, and 5-6). One thousand randomly sampled examples were used in these experiments.

A mutual nearest neighbor graph was first constructed using five nearest neighbors and then the graph Laplacian was formed. The elements of the weight matrix $\mathbf{W}$ were all binary. In the case of MNIST digits, raw pixel values (each feature normalized to mean zero and variance one) were used as features. For digits, nearest neighbors were determined by Euclidean distance whereas for text, nearest neighbors were determined by cosine similarity measure on tf-idf features. In the experiments, the number of eigenvalues q was set to 200. This was a uniform choice for all methods and for all the datasets which would not yield any unfair advantages for one method over any other. In the case of STORM and STOAM, ϵ was set to a negligible value of 10^{-6}.

The entire dataset was randomly divided into labeled and unlabeled examples. The

	l	MKL-S	MKL-R	SGT	KTA-S	KTA-R	STOAM	STORM
r-a	30	37.1 ±5.6	37.1 ±5.6	19.5 ±1.4	23.0 ±4.8	23.0 ±4.8	25.8 ±6.1	25.8 ±6.1
	50	29.9 ±5.1	30.0 ±5.2	18.9 ±1.1	19.9 ±3.1	19.9 ±3.1	21.5 ±4.0	21.5 ±4.0
	70	25.2 ±4.4	25.4 ±4.3	18.4 ±1.0	18.3 ±2.4	18.3 ±2.4	18.5 ±3.1	18.5 ±3.1
	90	22.3 ±3.3	22.7 ±3.3	18.2 ±0.9	17.3 ±1.5	17.3 ±1.5	17.2 ±1.8	17.2 ±1.9
	110	20.4 ±2.4	20.4 ±2.4	18.1 ±1.0	16.5 ±1.3	16.5 ±1.3	16.4 ±1.2	16.4 ±1.2
w-m	30	22.7 ±8.7	22.7 ±8.7	41.9 ±8.5	16.0 ±8.8	16.1 ±8.8	14.3 ±5.9	14.3 ±5.9
	50	15.1 ±3.8	15.1 ±3.8	35.6 ±9.3	13.5 ±3.4	13.6 ±3.4	11.5 ±3.4	11.5 ±3.4
	70	13.0 ±1.6	13.0 ±1.6	29.0 ±7.8	12.7 ±4.8	12.9 ±5.0	10.7 ±0.9	10.8 ±1.0
	90	12.2 ±1.6	12.2 ±1.6	22.5 ±6.3	11.3 ±1.5	11.4 ±1.7	10.4 ±0.6	10.4 ±0.6
	110	11.8 ±1.0	11.9 ±1.0	18.2 ±5.0	10.9 ±1.4	11.0 ±1.7	10.3 ±0.6	10.3 ±0.6
p-m	30	41.2 ±4.9	41.0 ±5.0	39.6 ±3.8	28.0 ±5.8	28.0 ±5.8	30.6 ±6.6	30.6 ±6.6
	50	36.0 ±5.3	35.9 ±4.9	37.5 ±3.8	24.3 ±4.8	24.3 ±4.8	25.7 ±4.6	25.7 ±4.6
	70	31.5 ±4.6	31.2 ±4.3	35.5 ±3.4	22.1 ±3.6	22.1 ±3.6	22.3 ±4.9	22.3 ±4.9
	90	28.1 ±3.8	28.3 ±3.8	33.6 ±3.4	20.6 ±2.8	20.6 ±2.7	20.4 ±3.0	20.8 ±3.2
	110	25.9 ±3.1	26.2 ±2.9	32.2 ±3.2	19.5 ±2.2	19.6 ±2.2	19.7 ±2.4	19.7 ±2.4
b-h	30	4.3 ±0.8	4.3 ±0.8	3.9 ±0.2	3.9 ±0.4	3.8 ±0.3	3.9 ±0.3	3.9 ±0.3
	50	3.9 ±0.1	3.9 ±0.1	3.9 ±0.2	3.8 ±0.3	3.8 ±0.4	3.9 ±0.3	3.7 ±0.3
	70	3.9 ±0.2	3.9 ±0.2	3.9 ±0.2	3.8 ±0.3	3.8 ±0.3	3.8 ±0.3	3.7 ±0.3
	90	3.9 ±0.2	3.9 ±0.2	3.9 ±0.3	3.7 ±0.3	3.7 ±0.3	3.8 ±0.3	3.6 ±0.3
	110	3.9 ±0.2	3.9 ±0.2	3.8 ±0.3	3.7 ±0.4	3.7 ±0.3	3.7 ±0.3	3.6 ±0.3
m-m	30	12.4 ±5.2	12.4 ±5.2	41.3 ±3.5	7.4 ±3.6	7.4 ±3.8	7.6 ±3.9	6.9 ±2.9
	50	7.5 ±3.1	7.3 ±2.9	31.2 ±7.5	6.3 ±2.8	6.2 ±2.9	5.5 ±1.0	5.4 ±1.2
	70	6.0 ±1.3	6.0 ±1.4	22.3 ±7.5	5.4 ±1.0	5.4 ±1.1	5.2 ±0.7	4.9 ±0.6
	90	5.7 ±1.0	5.7 ±1.0	15.4 ±5.9	5.1 ±0.9	5.1 ±1.1	5.1 ±0.6	4.8 ±0.6
	110	5.4 ±0.7	5.2 ±0.6	11.0 ±3.9	5.0 ±0.8	4.9 ±0.9	5.0 ±0.5	4.7 ±0.5

Table 5.1: Mean and std. deviation of percentage error rates on text datasets. In each row, the method with minimum error rate is shown in dark gray. All the other algorithms whose performance is not significantly different from the best (at 5% significance level by a paired t-test) are shown in light gray.

	l	MKL-S	MKL-R	SGT	KTA-S	KTA-R	STOAM	STORM
0-9	30	0.9 ±0.1	0.9 ±0.1	0.8 ±0.1	0.9 ±0.1	0.9 ±0.1	0.9 ±0.1	0.9 ±0.1
	50	0.9 ±0.1	0.9 ±0.1	0.8 ±0.1	0.9 ±0.1	0.9 ±0.1	0.9 ±0.1	0.9 ±0.1
	70	0.9 ±0.1	0.9 ±0.1	0.9 ±0.1	0.9 ±0.1	0.9 ±0.2	0.9 ±0.1	0.9 ±0.1
	90	0.9 ±0.1	0.9 ±0.2	0.9 ±0.1	0.9 ±0.2	0.9 ±0.2	0.9 ±0.1	0.9 ±0.1
	110	0.8 ±0.2	0.9 ±0.2	0.9 ±0.1	0.9 ±0.1	0.9 ±0.1	0.9 ±0.3	0.9 ±0.1
1-2	30	3.4 ±3.3	4.1 ±5.9	11.8 ±6.8	2.9 ±0.6	2.9 ±0.6	2.9 ±0.5	2.8 ±0.4
	50	2.9 ±0.5	2.6 ±0.4	3.6 ±2.7	2.8 ±0.4	2.8 ±0.5	2.8 ±0.7	2.8 ±0.7
	70	2.6 ±0.3	2.3 ±0.3	2.7 ±0.5	2.7 ±0.3	2.8 ±0.4	2.6 ±0.3	2.7 ±0.3
	90	2.7 ±0.3	2.3 ±0.3	2.6 ±0.2	2.8 ±0.3	2.7 ±0.4	2.7 ±0.4	2.7 ±0.3
	110	2.8 ±0.3	2.4 ±0.3	2.6 ±0.2	2.6 ±0.6	2.6 ±0.6	2.5 ±0.3	2.5 ±0.3
3-8	30	13.0 ±3.7	12.6 ±3.6	9.9 ±0.9	8.5 ±2.7	7.6 ±2.2	7.9 ±2.2	7.7 ±1.8
	50	9.5 ±2.3	9.0 ±2.2	8.8 ±0.9	6.9 ±1.8	6.6 ±1.6	6.4 ±1.5	6.4 ±1.4
	70	8.0 ±2.1	7.4 ±1.7	8.0 ±0.8	6.3 ±1.6	6.1 ±1.4	5.8 ±1.3	5.9 ±1.1
	90	7.0 ±1.6	6.6 ±1.3	7.3 ±0.8	5.7 ±1.1	5.7 ±1.1	5.5 ±1.0	5.4 ±0.9
	110	6.6 ±1.2	6.1 ±1.0	6.9 ±0.9	5.4 ±0.9	5.4 ±0.9	5.3 ±0.8	5.2 ±0.9
4-7	30	5.7 ±3.4	5.5 ±3.3	5.6 ±1.2	4.3 ±1.9	4.1 ±1.9	3.6 ±1.4	3.6 ±1.1
	50	4.3 ±1.2	4.0 ±0.9	4.5 ±0.5	3.5 ±0.9	3.4 ±0.8	3.2 ±0.7	3.2 ±0.6
	70	3.7 ±0.8	3.3 ±0.6	4.0 ±0.4	3.4 ±0.8	3.2 ±0.7	3.1 ±0.6	3.0 ±0.5
	90	3.5 ±0.8	3.1 ±0.6	3.8 ±0.4	3.1 ±0.6	3.0 ±0.6	2.9 ±0.5	2.9 ±0.5
	110	3.3 ±0.7	3.0 ±0.5	3.6 ±0.4	3.0 ±0.6	3.0 ±0.6	2.9 ±0.5	2.9 ±0.5
5-6	30	5.2 ±2.7	4.9 ±3.2	2.5 ±0.2	3.5 ±1.3	3.3 ±1.1	3.2 ±1.4	3.0 ±0.9
	50	3.3 ±1.3	2.9 ±0.8	2.5 ±0.2	2.9 ±0.7	2.9 ±0.5	2.7 ±0.4	2.7 ±0.4
	70	2.8 ±0.5	2.6 ±0.3	2.5 ±0.2	2.7 ±0.4	2.7 ±0.4	2.6 ±0.3	2.8 ±0.6
	90	2.7 ±0.3	2.6 ±0.3	2.5 ±0.2	2.6 ±0.4	2.6 ±0.4	2.6 ±0.3	2.5 ±0.4
	110	2.6 ±0.3	2.5 ±0.3	2.5 ±0.2	2.6 ±0.4	2.5 ±0.4	2.5 ±0.4	2.5 ±0.4

Table 5.2: Mean and std. deviation of percentage error rates on digits datasets. In each row, the method with minimum error rate is shown in dark gray. All the other algorithms whose performance is not significantly different from the best (at 5% significance level by a paired t-test) are shown in light gray.

	MKL-S	MKL-R	SGT	KTA-S	KTA-R	STOAM	STORM
#dark gray	1	5	9	5	2	8	22
#light gray	1	2	4	8	12	16	13
#total	2	7	13	13	14	24	35

Table 5.3: Summary of results in Tables 5.1 & 5.2. For each method, the number of times it performs the best (dark gray), the number of times it is not significantly worse from the best performing method (light gray) and the total number of times it is either the best or not significantly worse from the best are enumerated.

number of labeled examples was varied in steps of 20; the rest of the examples served as the test examples (as well as the unlabeled examples in graph construction). Kernel target alignment was then run to construct a kernel; the estimated kernel was then fed into an SVM (referred to KTA-S in the Tables) as well as to an RMM (referred to as KTA-R). To get an idea of the extent to which the ordering constraints matter, multiple kernel learning optimization was also performed; these are similar to the proposed formulations but without any ordering constraints on the weights. Multiple kernel learning with the RMM objective is referred to as MKL-R. Likewise, multiple kernel learning with the SVM objective is referred to as MKL-S. Similarly, experiments were also performed with the spectral graph transducer (referred to as SGT). Predictions on all the unlabeled examples were obtained from all the methods. Error rates were obtained on the unlabeled examples. Twenty such runs were done for various values of parameters (such as C,B) for all the methods. The values of the parameters that resulted in minimum average error rate over unlabeled examples were selected for all the approaches. Once the parameter values were fixed, the entire dataset was again divided into labeled and unlabeled examples. Training was then done but with fixed values of various parameters. Error rates on unlabeled examples were then obtained for all the methods over hundred runs of random splits of the dataset.

The results are presented in Tables 5.1 and 5.2. It can be seen that STORM and STOAM perform much better than all the methods. Results in the two tables are further summarized in Table 5.3. It can be seen that both STORM and STOAM have significant

advantage over all the other methods.

Learned spectra were visualized in each problem for various methods to see if there were any differences. Four such typical plots are presented in Figure 5.1. Spectra obtained by KTA, STORM and MKL-R are presented here. The spectra obtained by STOAM (MKL-S) was very close to the spectra obtained by STORM (MKL-R) compared to the other methods. Typically KTA puts significantly more weight on the top few eigenvectors. By not maintaining the order among the eigenvectors, MKL seems to put haphazard weights on the eigenvectors. However, STORM is less aggressive and its eigenspectrum decays at a slower rate. This shows that STORM obtains a markedly different spectrum compared to KTA and MKL thus resulting in a qualitatively different kernel. It is important to point out that MKL-R (MKL-S) solves a more general problem than STORM (STOAM). Thus, it can always achieve a better objective value compared to STORM (STOAM). However, the experiments show that the error rate on the unlabeled examples can be quite high without preserving the order on the spectrum. In fact, MKL obtained competitive results in only one case (digits:1-2).

5.6 Summary

A large relative margin formulation was proposed in this chapter for transforming the spectrum of a graph Laplacian. A family of kernels, which maintains smoothness properties on the graph, was explored by enforcing an ordering on the eigenvalues of the graph Laplacian. Unlike the previous methods which used two distinct criteria at each phase, this chapter demonstrated that it is possible to jointly optimize the spectrum as well as the classifier. The resulting kernels, learned as part of the optimization, showed improvements on a variety of experiments. The relative margin criterion was shown to perform much better compared to large absolute margin criterion.

This work opens up an interesting research direction for further investigation: by learning weights on an appropriate number of matrices, we can potentially explore all positive-semidefinite matrices. It is thus possible to explore all graphs while learning a large relative margin classifier.

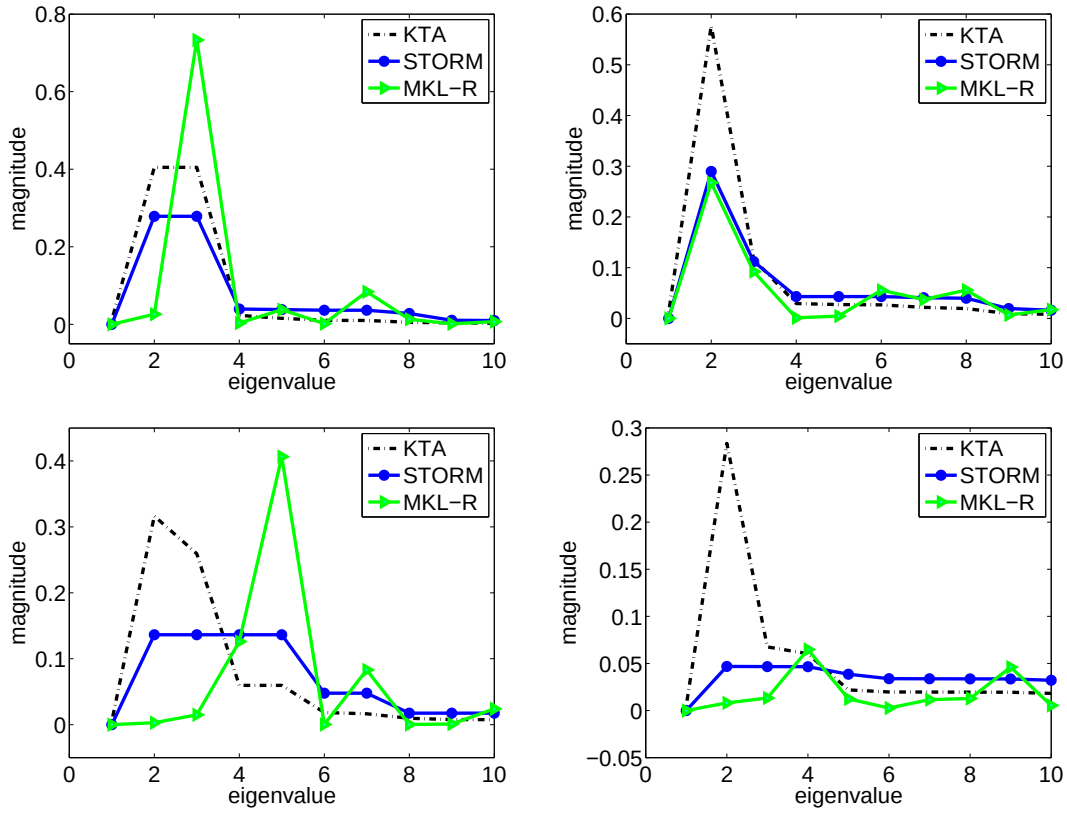

Figure 5.1: Magnitude of the top 15 eigenvalues as learned by different algorithms. Top: problems 1-2 and 3-8. Bottom: m-m and p-m. The plots show average eigenspectra over all runs for each problem.

Chapter 6

Boosting

The aim of this chapter is to develop a boosting algorithm based on the ideas of relative margin. While the development of algorithms in the previous chapters mainly relied on affine transformation arguments, a completely different motivation will be provided for the boosting algorithm developed in this chapter. The main idea is to trade-off between the exponential loss on the training examples and its variance. The motivation for this will be provided from the recent empirical Bernstein bounds. This work was first published in [Shivaswamy and Jebara, 2010b].

In classification problems, many machine learning algorithms minimize a convex upper bound on the misclassification rate. For example, AdaBoost [Freund and Schapire, 1997] minimizes the exponential loss and support vector machines [Vapnik, 1995] minimize the hinge loss. The convexity of such losses is helpful for computational as well as generalization reasons [Bartlett *et al.*, 2006]. In most problems, the aim is not to obtain a function that performs well on training data but rather to estimate a function (using training data) that performs well on future unseen test data. This is accomplished by minimizing empirical risk on the training set while choosing a function of small complexity. The rationale behind this approach is that the empirical risk converges (uniformly) to the true unknown risk. Various concentration inequalities show how fast the empirical risk converges to the true risk.

A key tool in obtaining such bounds is Hoeffding's inequality which relates the empirical mean of a bounded random variable to its true mean. Bernstein's and Bennett's inequalities relate the true mean of a random variable to the empirical mean but also incorporate the

true variance of the random variable. If the true variance of a random variable is small, these bounds can be significantly tighter than Hoeffding's bound. Recently, there have been empirical counterparts of Bernstein's inequality [Audibert *et al.*, 2007; Maurer and Pontil, 2009]; these bounds incorporate the empirical variance of a random variable rather than its true variance. The advantage of these bounds is that the quantities they involve are actually observable. Previously, these bounds have been applied in sampling procedures [Mnih *et al.*, 2008]. In this chapter, a new boosting algorithm that not only minimizes the empirical misclassification rate but also the empirical variance on the exponential loss is proposed. This is achieved via the so called "sample variance penalization" principle which is motivated from the empirical Bernstein inequality. The proposed algorithm is easy to implement and involves a closed form update rule like the AdaBoost algorithm.

The rest of this chapter is organized as follows: various bounds and their uniform convergence counterparts are reviewed in Section 6.1. Sample variance penalization with the 0–1 loss is then shown to be equivalent to empirical risk minimization. Subsequently, sample variance penalization principle is applied on the exponential loss whereby a novel boosting algorithm is obtained in Section 6.3. Experiments are then presented on a number of datasets in Section 6.4 followed by a discussion in Section 6.4.1. A summary of this chapter is provided in Section 6.5.

6.1 Hoeffding and empirical Bernstein bounds

In this section, some classical as well as new concentration inequalities are reviewed. The purpose of this section is to present sample variance penalization [Maurer and Pontil, 2009] as a motivation for the rest of the paper. However, for the sake of clarity and completeness, a few other classic results are reviewed as well. First, recall Hoeffding's inequality.

Theorem 26 *Let* $\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_n$ *be* iid *random variables with values in* $[0, 1]$*. Then, for any* $\delta > 0$*, with probability at least* $1-\delta$ *(over the draw of* $\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_n$*), the following inequality holds:*

$$\mathbf{E}[\mathbf{z}] \leq \frac{1}{n}\sum_{i=1}^{n} \mathbf{z}_i + \sqrt{\frac{\ln(1/\delta)}{2n}}.$$

In machine learning settings, training examples $(\mathbf{x}_i, y_i)_{i=1}^n$ are drawn *iid* from a distribution $\mathcal{D}$ where $(\mathbf{x}_i, y_i) \in \mathcal{X} \times \mathcal{Y}$. From this data, a learning algorithm outputs a function $f : \mathcal{X} \to \mathbb{R}$ (typically from a fixed set of functions) that should predict well on future samples. In other words, the function should minimize a fixed loss $l : \mathbb{R} \times \mathcal{Y} \to [0, 1]$ on unseen test examples drawn from the same distribution.

Empirical risk minimization (ERM) is a popular approach to minimizing loss on unseen test examples. Suppose a learning algorithm is to choose a function from a finite set of candidate functions $\mathcal{F}$. Consider the extension of Hoeffding's inequality such that it holds uniformly over all functions in $\mathcal{F}$.

Corollary 27 *Let $(\mathbf{x}_i, y_i)_{i=1}^n$ be drawn* iid *from a distribution $\mathcal{D}$. Let $\mathcal{F}$ be a finite class of functions $f : \mathcal{X} \to \mathbb{R}$. Then, given a loss function $l : \mathbb{R} \times \mathcal{Y} \to [0, 1]$, for any $\delta > 0$, with probability at least $1 - \delta$, $\forall f \in \mathcal{F}$,*

$$\mathbf{E}[l(f(\mathbf{x}), y)] \leq \frac{1}{n} \sum_{i=1}^{n} l(f(\mathbf{x}_i), y_i) + \sqrt{\frac{\ln(|\mathcal{F}|)/\delta}{2n}} \tag{6.1}$$

where $|\mathcal{F}|$ is the cardinality of $\mathcal{F}$.

In fact, the above corollary can be extended to infinite function classes $\mathcal{F}$ by replacing the term $|\mathcal{F}|$ with a suitable complexity measure. The empirical risk minimization principle selects a function from a class to minimize empirical loss on the training data, *i.e.,*

$$\arg\min_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^{n} l(f(\mathbf{x}_i), y_i).$$

Since (6.1) holds uniformly over all the functions in the class $\mathcal{F}$, minimizing the empirical risk minimizes an upper bound on the future loss.

As an alternative to Hoeffding's bound, consider Bernstein's inequality.

Theorem 28 *Under the same conditions as Theorem 26, for any $\delta > 0$, with probability at least $1 - \delta$,*

$$\mathbf{E}[\mathbf{z}] \leq \frac{1}{n} \sum_{i=1}^{n} \mathbf{z}_i + \sqrt{\frac{2\mathbf{V}[\mathbf{z}] \ln(1/\delta)}{n}} + \frac{\ln(1/\delta)}{3n},$$

where $\mathbf{V}[\mathbf{z}] = \mathbf{E}\left(\mathbf{z} - \mathbf{E}[\mathbf{z}]\right)^2$.

When the variance $\mathbf{V}[\mathbf{z}]$ of the random variable $\mathbf{z}$ is small, the above bound can be significantly tighter than Hoeffding's bound. To get an idea of why the bound in Theorem 28 can be better than the one in Theorem 26, consider a situation in which $\mathbf{V}[\mathbf{z}] = 0$. In this scenario, $\frac{1}{n}\sum_{i=1}^{n} \mathbf{z}_i$ converges to $\mathbf{E}[\mathbf{z}]$ at the rate $\mathcal{O}(1/n)$ according to Bernstein's inequality. However, in the case of Hoeffding's inequality, the convergence is at a much slower $\mathcal{O}(1/\sqrt{n})$ rate. However, one limitation of the above bound is that the true value $\mathbf{V}[\mathbf{z}]$ is often an unknown quantity and only an empirical estimate is available. To address this limitation, recall the following result from [Maurer and Pontil, 2009] which is similar to Theorem 28 but holds for an empirical estimate of the variance as opposed to the true value $\mathbf{V}[\mathbf{z}]$.

Theorem 29 *Under the same conditions as Theorem 26, for any $\delta > 0$, with probability at least $1 - \delta$,*

$$\mathbf{E}[\mathbf{z}] \leq \frac{1}{n}\sum_{i=1}^{n} \mathbf{z}_i + \sqrt{\frac{2\hat{\mathbf{V}}[\mathbf{z}]\ln(2/\delta)}{n}} + \frac{7\ln(2/\delta)}{3(n-1)},$$

where $\hat{\mathbf{V}}[\mathbf{z}]$ is the empirical variance given by:[1]

$$\hat{\mathbf{V}}[\mathbf{z}] = \frac{1}{n(n-1)} \sum_{n \geq i > j \geq 1} (\mathbf{z}_i - \mathbf{z}_j)^2.$$

The above theorem has the advantage that all the quantities involved are empirical quantities that can be obtained from data. Finally, the following uniform convergence extension of the above bound serves as the motivation for sample variance penalization.

Theorem 30 *Let $(\mathbf{x}_i, y_i)_{i=1}^{n}$ be drawn* iid *from a distribution $\mathcal{D}$. Let $\mathcal{F}$ be a class of functions $f : \mathbf{x} \to \mathbb{R}$. Then, given a loss function $l : \mathbb{R} \times \mathcal{Y} \to [0, 1]$, for any $\delta > 0$, with probability at least $1 - \delta$, $\forall f \in \mathcal{F}$,*

$$\mathbf{E}[l(f(\mathbf{x}), y)] \leq \frac{1}{n}\sum_{i=1}^{n} l(f(\mathbf{x}_i), y_i) + \frac{15\ln(\mathcal{M}(n)/\delta)}{(n-1)} + \sqrt{\frac{18\hat{\mathbf{V}}[l(f(\mathbf{x}), y)]\ln(\mathcal{M}(n)/\delta)}{n}}, \quad (6.2)$$

where $\mathcal{M}(n)$ is a complexity measure [Maurer and Pontil, 2009]. Unlike the empirical risk in (6.2), the empirical variance $\hat{\mathbf{V}}[l(f(\mathbf{x}), y)]$ has a multiplicative factor involving $\mathcal{M}(n)$ (which is typically difficult to estimate), δ (the required confidence at which the bound holds) and

[1] For brevity, unless extra constraints on i and j are specified, $i > j$ in future summations in this chapter must be read as $n \geq i > j \geq 1$.

n. Thus, for a given problem, it is difficult to specify the trade-off between empirical risk and empirical variance a priori. Hence, the algorithm proposed in this chapter will necessarily involve a scalar parameter which trades off between these two criteria. As is often the case in practice, this trade-off parameter has to be tuned using a validation set. Minimizing this uniform convergence bound leads to the so-called sample variance penalization principle:

$$\arg\min_{f\in\mathcal{F}} \frac{1}{n}\sum_{i=1}^{n} l(f(\mathbf{x}_i), y_i) + \lambda\sqrt{\frac{\hat{\mathbf{V}}[l(f(\mathbf{x}), y)]}{n}},$$

where $\lambda \geq 0$ trades off between the empirical risk and the empirical variance. The aim of the rest of this chapter is to derive an efficient, AdaBoost style, algorithm for sample variance penalization.

6.2 Loss functions

In this section, first, the possibility of sample variance penalization with the $0-1$ loss is considered. It turns out that this merely corresponds to empirical risk minimization. Subsequently, the sample variance penalization principle is applied to the exponential loss where it produces a qualitatively different criterion.

In the previous section, generic loss functions—without any assumptions on the problem type—were studied. However, from now on, attention will be restricted to binary classification problems. In classification problems $\mathcal{Y} = \{\pm 1\}$. Define the classification loss function $l_1(f(\mathbf{x}), y)$ as:

$$l_1(f(\mathbf{x}), y) := \mathbf{I}_{yf(\mathbf{x})\leq 0} = \begin{cases} 0 & \text{if } yf(\mathbf{x}) > 0, \\ 1 & \text{if } yf(\mathbf{x}) \leq 0. \end{cases} \tag{6.3}$$

Given an *iid* set of examples $(\mathbf{x}_i, y_i)_{i=1}^n$, the aim is to minimize the future probability of error, *i.e.,* to minimize $\mathbf{P}_{\mathcal{D}}[yf(\mathbf{x}) \leq 0]$.

Lemma 31 *Let $(\mathbf{x}_i, y_i)_{i=1}^n$ be drawn* iid *from a distribution $\mathcal{D}$. Let $\mathcal{F}$ be a class of functions $f : \mathbf{x} \to \mathcal{Y}$. Then, for any $\delta > 0$, with probability at least $1-\delta$, $\forall f \in \mathcal{F}$,*

$$\mathbf{P}_{\mathcal{D}}[yf(\mathbf{x}) \leq 0] \leq \hat{p} + \sqrt{\frac{18\hat{p}(1-\hat{p})\ln(\mathcal{M}(n)/\delta)}{n-1}} + \frac{15\ln(\mathcal{M}(n)/\delta)}{(n-1)}, \tag{6.4}$$

where $\hat{p} = \hat{\mathbf{P}}[l_1(f(\mathbf{x}), y)] = \frac{1}{n}\sum_{i=1}^{n} \mathbf{I}_{y_i f(\mathbf{x}_i)\leq 0}$ is the empirical error.

Proof The proof is a direct application of Theorem (30) on the 0-1 loss (6.3). First observe that,

$$\mathbf{E}_{\mathcal{D}}[\mathbf{I}_{yf(\mathbf{x})\leq 0}] = \mathbf{P}_{\mathcal{D}}[yf(\mathbf{x}) \leq 0].$$

Moreover, denoting $\mathbf{I}_{y_i f(\mathbf{x}_i)\leq 0}$ by s_i for brevity,

$$\begin{aligned}
\hat{\mathbf{V}}[l_1(f(\mathbf{x}), y)] &= \frac{1}{n(n-1)} \sum_{i>j} (s_i - s_j)^2 \\
&= \frac{1}{n(n-1)} \left((n-1) \sum_{i=1}^{n} s_i^2 - 2 \sum_{i>j} s_i s_j \right) \\
&= \frac{n}{n-1} \left(\frac{1}{n} \sum_{i=1}^{n} s_i^2 - \left(\frac{1}{n} \sum_{i=1}^{n} s_i \right)^2 \right) \\
&= \frac{n}{n-1} \hat{p}(1-\hat{p}).
\end{aligned}$$

For an indicator random variable, $s_i^2 = s_i$. This fact was used in going from line three to four. ■

Thus, to minimize future classification error, sample variance penalization suggests minimizing $\hat{p} + \lambda\sqrt{\hat{p}(1-\hat{p})}$. Consider $\hat{p} \in [0, 1/2)$ which is clearly the regime of interest in classification problems. It is easy to see that, for $\hat{p} \in [0, 1/2)$, the quantity $\hat{p} + \lambda\sqrt{\hat{p}(1-\hat{p})}$ is a monotonically increasing function of the empirical error $\hat{p}$. Therefore, sample variance penalization is equivalent to minimizing the empirical error $\hat{p}$. Thus, for any finite non-negative value of λ, sample variance penalization merely reduces to empirical risk minimization with the $0-1$ loss.

6.2.1 Minimizing a convex upper bound on the $0-1$ loss

It is important to note that empirical risk minimization with the $0-1$ loss is a hard problem; this problem is often circumvented by minimizing convex upper bounds on this empirical risk. In this chapter, the following exponential loss is considered:

$$l_2(f(\mathbf{x}), y) := e^{-yf(\mathbf{x})}. \tag{6.5}$$

Consider functions $f : \mathbf{x} \to [-1, 1]$. The aim here is to still minimize the future $0-1$ loss yet only through the surrogate exponential loss function. A computational advantage of this convex loss function is that it is easy to minimize. First, the future probability of error is related to the exponential loss as below:

$$\mathbf{P}_{\mathcal{D}}[yf(\mathbf{x}) \leq 0] = \mathbf{E}_{\mathcal{D}}[\mathbf{I}_{yf(\mathbf{x})\leq 0}] \leq \mathbf{E}_{\mathcal{D}}[e^{-yf(\mathbf{x})}].$$

It is now possible to relate the future probability of error to the empirical mean and the empirical variance of the exponential loss. By applying the result from Theorem 30:

$$\mathbf{E}_{\mathcal{D}}[e^{-yf(\mathbf{x})}] \leq \frac{1}{n}\sum_{i=1}^{n} e^{-y_i f(\mathbf{x}_i)} + \frac{15e\ln(\mathcal{M}(n)/\delta)}{(n-1)} + \sqrt{\frac{\sum_{i>j}(e^{-y_i f(\mathbf{x}_i)} - e^{-y_j f(\mathbf{x}_j)})^2}{n(n-1)}}\sqrt{\frac{18\ln(\mathcal{M}(n)/\delta)}{n}},$$

where an extra e appears in the second term to normalize the exponential loss so that it has the range $[0, 1]$ (such that Theorem 30 applies directly).

Thus, the sample variance penalization principle, applied to the exponential loss suggests minimizing the following quantity for some scalar $\tau > 0$:

$$\sum_{i=1}^{n} e^{-y_i f(\mathbf{x}_i)} + \tau\sqrt{\sum_{i>j}\left(e^{-y_i f(\mathbf{x}_i)} - e^{-y_j f(\mathbf{x}_j)}\right)^2}. \tag{6.6}$$

6.3 A boosting algorithm

In this section, a boosting style algorithm to minimize (6.6) is derived. Assume the function class $\mathcal{H}$ of interest is composed of so-called weak learners $G^s : \mathcal{X} \to \{\pm 1\}$ for $s = 1, \ldots, S$. The function to be output consists of the following additive model over weak learners:

$$f(\mathbf{x}) = \sum_{s=1}^{S} \alpha_s G^s(\mathbf{x}). \tag{6.7}$$

where $\alpha_s \in \mathbb{R}^+$ for $s = 1, \ldots, S$.

One minor technicality is that the analysis in the previous section assumed that the function f had a range $[-1, 1]$. However, while building an additive model (6.7) this does not hold. Thus the range of the function obtained both by AdaBoost and the proposed algorithm will be $[e^{-\sum_{s=1}^{S}\alpha_s}, e^{\sum_{s=1}^{S}\alpha_s}]$.

Algorithm 6.1 AdaBoost

Require: $(X_i, y_i)_{i=1}^n$, weak learners $\mathcal{H}$

Initialize the weights: $w_i \leftarrow \frac{1}{n}$; initialize f to predict zero on all inputs.

for $s \leftarrow 1$ to S **do**

Get a weak learner $G^s(\cdot)$ that minimizes $\sum_{i:y_i \neq G^s(X_i)} w_i$

$\alpha_s = \frac{1}{2} \log \left(\frac{\sum_{y_i = G^s(X_i)} w_i}{\sum_{y_i \neq G^s(X_i)} w_i} \right)$

if $\alpha_s < 0$ **then**

break

end if

$f(\cdot) \leftarrow f(\cdot) + \alpha_s G^s(\cdot)$

$w_i \leftarrow w_i \exp(-y_i G^s(X_i) \alpha_s)/Z_s$ where Z_s is such that $\sum_{i=1}^n w_i = 1$.

end for

6.3.1 AdaBoost

AdaBoost is described in **Algorithm** 6.1. Since it is a well studied algorithm, further details are not provided. AdaBoost merely minimizes an exponential loss, *i.e.,* it minimizes $\sum_{i=1}^n e^{-y_i f(\mathbf{x}_i)}$ in a stage-wise manner to build an additive model (6.7).

6.3.2 An update rule for empirical Bernstein boosting

The update rule in this chapter is based on the stage-wise greedy optimization interpretation of AdaBoost [Hastie *et al.*, 2001]. Sample variance penalization will be performed on the exponential loss; an additive model (6.7) will be built in the process. The update for the proposed boosting algorithm is based on the stage-wise greedy interpretation of AdaBoost [Hastie *et al.*, 2001]. The aim is to minimize the sample variance cost (6.6) while building an additive model (6.7). As in most boosting methods, a convex cost is minimized in each stage of the algorithm. Effectively, the same class of weak learners as AdaBoost (*i.e.*, a conic combination of weak learners) is considered while performing sample variance penalization instead of empirical risk minimization.

Since there is an unknown trade-off between the two terms in (6.6), one way to minimize

that cost is by the following minimization:

$$\min_{f\in\mathcal{F}} \sum_{i=1}^{n} e^{-y_i f(\mathbf{x}_i)}$$
$$\text{s.t.} \sqrt{\sum_{i>j} \left(e^{-y_i f(\mathbf{x}_i)} - e^{-y_j f(\mathbf{x}_j)}\right)^2} \leq B,$$

where the trade-off parameter is now parametrized by B. For every value of τ there is a B that obtains the same optimal function; in particular, $B = \infty$ is equivalent to $\tau = 0$. The above problem can be equivalently posed as:

$$\min_{f\in\mathcal{F}} \left(\sum_{i=1}^{n} e^{-y_i f(\mathbf{x}_i)}\right)^2$$
$$\text{s.t.} \sum_{i>j} \left(e^{-y_i f(\mathbf{x}_i)} - e^{-y_j f(\mathbf{x}_j)}\right)^2 \leq B^2.$$

By introducing a non-negative Lagrange multiplier, the above optimization problem can be written as:

$$\left(\sum_{i=1}^{n} e^{-y_i f(\mathbf{x}_i)}\right)^2 + \lambda \sum_{i>j} \left(e^{-y_i f(\mathbf{x}_i)} - e^{-y_j f(\mathbf{x}_j)}\right)^2 - \lambda B^2.$$

One way to implement the above optimization is to minimize it under different settings of λ. Since the last term does not involve the function f, the following objective is optimized:

$$\min_{f\in\mathcal{F}} \left(\sum_{i=1}^{n} e^{-y_i f(\mathbf{x}_i)}\right)^2 + \lambda \sum_{i>j} \left(e^{-y_i f(\mathbf{x}_i)} - e^{-y_j f(\mathbf{x}_j)}\right)^2. \tag{6.8}$$

It is assumed that there is already a function $h(\cdot)$; a greedy algorithm will then be derived to to obtain a weak learner $G(\cdot)$ and a positive scalar α such that $f(\cdot) = h(\cdot) + \alpha G(\cdot)$ minimizes the above objective the most.

Denoting $e^{-y_i h(\mathbf{x}_i)}$ by w_i, (6.8) can be written as:[2]

$$\left(\sum_{i=1}^{n} w_i e^{-y_i \alpha G(\mathbf{x}_i)}\right)^2 + \lambda \sum_{i>j} \left(w_i e^{-y_i \alpha G(\mathbf{x}_i)} - w_j e^{-y_j \alpha G(\mathbf{x}_j)}\right)^2$$
$$= (1 + (n-1)\lambda) \sum_{i=1}^{n} w_i^2 e^{-2y_i \alpha G(\mathbf{x}_i)} + (2 - 2\lambda) \sum_{i>j} w_i w_j e^{-y_i \alpha G(\mathbf{x}_i) - y_j \alpha G(\mathbf{x}_j)}. \tag{6.9}$$

[2]In the final algorithm there will be a normalization factor dividing w_i.

Algorithm 6.2 EBBoost

Require: $(\mathbf{x}_i, y_i)_{i=1}^n$, scalar parameter $\lambda \geq 0$, weak learners $\mathcal{H}$

Initialize the weights: $w_i \leftarrow \frac{1}{n}$; initialize f to predict 0 on all inputs.

for $s \leftarrow 1$ to S **do**

Get a weak learner $G^s(\cdot)$ that minimizes (6.9) with the following choice of α_s:

$\alpha_s = \frac{1}{4}\log\left(\frac{(1-\lambda)(\sum_{i\in I} w_i)^2 + \lambda n \sum_{i\in I} w_i^2}{(1-\lambda)(\sum_{i\in J} w_i)^2 + \lambda n \sum_{i\in J} w_i^2}\right)$

if $\alpha_s < 0$ **then** break **end if**

$f(\cdot) \leftarrow f(\cdot) + \alpha_s G^s(\cdot)$

$w_i \leftarrow w_i \exp(-y_i G^s(\mathbf{x}_i)\alpha_s)/\mathbf{z}_s$ where $\mathbf{z}_s$ is such that $\sum_{i=1}^n w_i = 1$.

end for

For brevity, define the following sets of indices:

$$I = \{i : y_i G(\mathbf{x}_i) = +1\}, \quad J = \{i : y_i G(\mathbf{x}_i) = -1\}.$$

Here, I denotes the set of examples that are correctly classified by $G(\cdot)$ and J is the set of misclassified examples. Equation (6.9) can now be rewritten as:

$$\lambda_1 \left(\sum_{i\in I} w_i^2 e^{-2\alpha} + \sum_{i\in J} w_i^2 e^{2\alpha}\right) + \lambda_2 \sum_{i>j:i,j\in I} w_i w_j e^{-2\alpha}$$
$$+ \lambda_2 \sum_{i>j:i,j\in J} w_i w_j e^{+2\alpha} + \lambda_2 \sum_{i>j:i\in I,j\in J \text{ or } i\in J,j\in I} w_i w_j,$$

where λ_1 and λ_2 are defined as $\lambda_1 = (1 + (n-1)\lambda)$ and $\lambda_2 = 2 - 2\lambda$. The above expression is convex in α; it is easy to see this by taking the second derivative with respect to α. The above expression is minimized by differentiating with respect to α and equating to zero:

$$\alpha = \frac{1}{4}\log\left(\frac{\lambda_1 \sum_{i\in I} w_i^2 + \lambda_2 \sum_{i>j:i,j\in I} w_i w_j}{\lambda_1 \sum_{i\in J} w_i^2 + \lambda_2 \sum_{i>j:i,j\in J} w_i w_j}\right).$$

At this point, it appears that all pairwise interactions between weights are needed to find α which would make the computation of α (given a weak learner G) cumbersome because of the $\mathcal{O}(n^2)$ pairwise terms. Consider the numerator in the update rule for α and substitute

the values of λ_1 and λ_2, to get

$$\begin{aligned}
&(1+(n-1)\lambda)\sum_{i\in I} w_i^2 + (2-2\lambda)\sum_{i>j:i,j\in I} w_i w_j \\
&= \sum_{i\in I} w_i^2 + 2\sum_{i>j:i,j\in I} w_i w_j + \lambda n \sum_{i\in I} w_i^2 - \lambda\left(\sum_{i\in I} w_i^2 + 2\sum_{i>j:i,j\in I} w_i w_j\right) \\
&= (1-\lambda)(\sum_{i\in I} w_i)^2 + \lambda n \sum_{i\in I} w_i^2.
\end{aligned}$$

Applying a similar simplification to the denominator yields the following $\mathcal{O}(n)$ rule

$$\alpha = \frac{1}{4}\log\left(\frac{(1-\lambda)(\sum_{i\in I} w_i)^2 + \lambda n \sum_{i\in I} w_i^2}{(1-\lambda)(\sum_{i\in J} w_i)^2 + \lambda n \sum_{i\in J} w_i^2}\right).$$

The algorithm based on sample variance penalization (6.6) is stated in **Algorithm** 6.2. It merely requires the sum of weights on examples and the sum of squared weights on appropriate partitions defined by the weak learner. Further, given a weak learner, note that (6.9) only requires $\mathcal{O}(n)$ time to evaluate.

It is easy to see that AdaBoost is a specific instance of EBBoost algorithm. For the choice $\lambda = 0$, (6.6) becomes $\left(\sum_{i=1}^n e^{-y_i f(\mathbf{x}_i)}\right)^2$. Even though this cost function is the AdaBoost cost squared, the optimal choice of α remains the same since the AdaBoost cost is non-negative. Substituting, $\lambda = 0$ in the expression for α above,

$$\alpha = \frac{1}{4}\log\left(\frac{(\sum_{i\in I} w_i)^2}{(\sum_{i\in J} w_i)^2}\right) = \frac{1}{2}\log\left(\frac{\sum_{i\in I} w_i}{\sum_{i\in J} w_i}\right),$$

which coincides with the choice of α in AdaBoost.

6.4 Experiments

In this section, the empirical performance of the proposed algorithm is studied. The primary comparison is between AdaBoost and EBBoost. There are numerous variants of boosting algorithms; many of these variants are exploring other intricacies of the classification problems (sparsity, robustness to outliers and so forth). While performance is reported for three variants, the goal is not to find a method that outperforms every variant of boosting under every possible choice of weak learners. In fact, many of these boosting variants could also be modified to incorporate sample variance penalization. The emphasis, rather, is on

comparing AdaBoost with EBBoost with a simple family of weak learners such as decision stumps. Experiments with three other boosting variants are included simply to give a broader perspective. The following variants of the boosting algorithms are considered in the experiments in this chapter:

Dataset	AdaBoost	EBBoost	RLP-Boost	RQP-Boost	ABR
a5a	18.07 ± 0.6	17.82 ± 0.6	17.90 ± 0.8	18.06 ± 0.9	17.80 ± 0.5
abalone	22.53 ± 0.8	22.38 ± 0.9	23.68 ± 1.3	23.01 ± 1.3	22.40 ± 0.7
image	4.28 ± 0.8	4.04 ± 0.8	4.19 ± 0.8	3.79 ± 0.7	4.27 ± 0.8
nist09	1.28 ± 0.2	1.17 ± 0.1	1.43 ± 0.2	1.25 ± 0.2	1.18 ± 0.2
nist14	0.80 ± 0.2	0.70 ± 0.1	0.89 ± 0.2	0.78 ± 0.2	0.74 ± 0.1
nist27	2.56 ± 0.3	2.41 ± 0.3	2.72 ± 0.3	2.49 ± 0.3	2.32 ± 0.3
nist38	5.68 ± 0.6	5.34 ± 0.4	6.04 ± 0.4	5.48 ± 0.5	5.24 ± 0.5
nist56	3.64 ± 0.5	3.38 ± 0.4	3.97 ± 0.5	3.61 ± 0.4	3.42 ± 0.3
mushrooms	0.35 ± 0.3	0.28 ± 0.3	0.30 ± 0.3	0.30 ± 0.3	0.29 ± 0.4
musklarge	7.80 ± 1.0	6.89 ± 0.6	7.83 ± 1.0	7.29 ± 1.0	7.22 ± 0.7
ringnorm	15.05 ± 3.1	13.45 ± 2.4	15.25 ± 4.2	14.55 ± 3.0	14.35 ± 3.1
spambase	7.74 ± 0.7	7.18 ± 0.8	7.45 ± 0.6	7.25 ± 0.7	6.99 ± 0.6
splice	10.57 ± 1.1	10.27 ± 0.9	10.28 ± 0.8	10.18 ± 1.0	10.02 ± 0.9
twonorm	4.30 ± 0.4	4.00 ± 0.2	4.87 ± 0.5	4.19 ± 0.4	4.16 ± 0.4
w4a	2.80 ± 0.2	2.75 ± 0.2	2.76 ± 0.1	2.77 ± 0.2	2.75 ± 0.2
waveform	12.96 ± 0.8	12.90 ± 0.8	12.75 ± 0.9	12.22 ± 0.9	12.47 ± 0.7
wine	26.03 ± 1.2	25.66 ± 1.0	25.00 ± 1.2	25.20 ± 1.0	25.09 ± 1.2
wisc	5.00 ± 1.5	4.00 ± 1.3	4.14 ± 1.5	4.71 ± 1.5	4.46 ± 1.6

Table 6.1: For each dataset, the algorithm with the best percentage test error is represented by a dark gray cell. All the light gray in a row denote results that are not significantly different from the minimum error (by a paired t-test at 5% significance level). EBBoost outperforms AdaBoost on all datasets.

Regularized LP and QP Boost Given a dataset, first AdaBoost is run to obtain training predictions from weak learners. LP and QP Boost algorithms [Raetsch *et al.*, 2001] then optimize the weights on weak learners obtained by AdaBoost to maximize the margin (along with regularization on the weights). Once the optimizations are solved, predictions are obtained based on the outputs of the same weak learners on the test set.

Boosting with Soft Margin $\text{AdaBoost}_{\text{REG}}$ (ABR) [Raetsch *et al.*, 2001] optimizes a "soft margin" by allowing slacks on examples; this is better suited to handle noisy situations.

Experiments were performed on a number of publicly available datasets. Only datasets, that had at least 400 examples so that both validation and test sets had at least 100 examples, were chosen. For each dataset, the minimum of half of the examples (or 500 examples) were chosen as training examples. This was done since solving an LP or QP with a large number of examples can be quite expensive compared to boosting. Moreover, ABR requires a line search which can also be much slower than AdaBoost. The remaining examples in each dataset were divided equally into validation and test sets by a random split. For AdaBoost, EBBoost, and ABR, 500 randomly generated decision stumps were considered as the weak learners. Each algorithm was run until there was no drop in the validation error rate in 50 iterations; the corresponding test error rate was then noted. The set of weak learners recovered by AdaBoost was given to regularized LP and QP boosting procedures. For all methods (other than AdaBoost) there is an extra parameter to tune. The value of the parameter that resulted in the minimum error on the validation set was used to obtain the test error. The experiment was repeated 20 times over random splits of training, test, and validation sets. Results are reported in Table 6.1.

EBBoost shows significant improvement over AdaBoost on most of the datasets; in fact, it shows an improvement over every single dataset. ABR's performance comes closest to EBBoost even though the methods are qualitatively quite different. In fact, it is straightforward to obtain a soft margin version of EBBoost by replacing the choice of loss function. Moreover, in the following section, it will be shown that the performance gains of EBBoost and ABR emerge for completely different reasons and the intuitions underlying the two may be complementary.

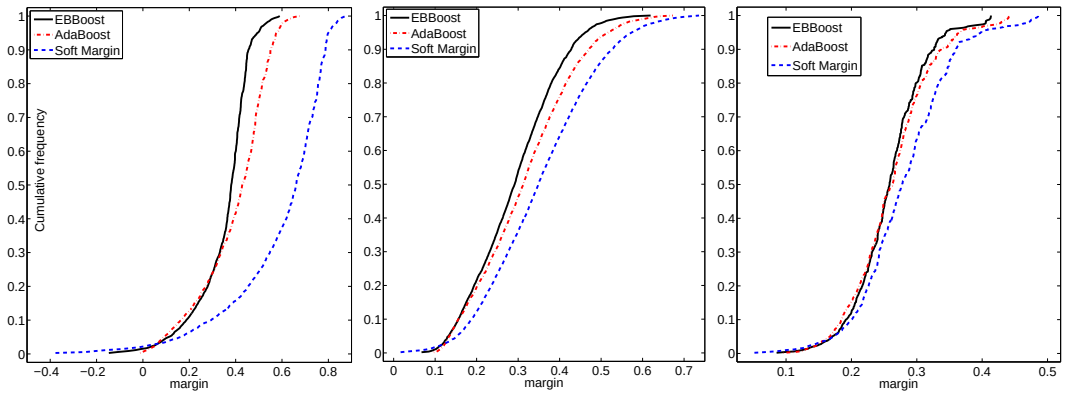

Figure 6.1: Cumulative margin distributions on three different datasets (wisconsin, mnist27, mushrooms). ABR obtains a long tail indicating its "slackness". EBBoost's margins are characterized by a smaller variance.

6.4.1 Discussion

Since EBBoost and ABR showed similar performance overall, it is interesting to see how the solutions differ. For this purpose, the margin distribution on the training examples were compared. The effectiveness of boosting can be (to some extent), explained by the margin distribution [Schapire *et al.*, 1998; Koltchinskii and Panchenko, 2002]. Recall the definition of margin on an example $\mathbf{x}_i$: $\gamma(\mathbf{x}_i, y_i) = y_i \sum_{s=1}^{S} \alpha_s G^s(\mathbf{x}_i) / \sum_{s=1}^{S} \alpha_s$, based on the additive model (6.7).

The margin distributions obtained by various algorithms on all the datasets were visualized. These plots show the average margin distribution over the experiments at the setting of the parameters selected by validation. Three typical cumulative margin distribution plots are shown in Figure 6.1. Even though ABR and EBBoost showed similar test error rates, they have fairly different margin distributions. Typically, ABR has a long tail over incorrect predictions (due to its use of slack on hard to classify examples) whereas EBBoost is characterized by a small variance (not surprisingly, since the variance with an exponential loss is being minimized). In addition, the mean and standard deviation of all the margin values on all datasets were obtained. Table 6.2 summarizes those results. ABR obtains larger mean margin as well as large standard deviations. EBBoost typically obtains

	AdaBoost	EBBoost	ABR
a5a	0.21 ± 0.20	0.19 ± 0.17	0.20 ± 0.19
abalone	0.12 ± 0.12	0.12 ± 0.12	0.13 ± 0.13
image	0.14 ± 0.08	0.13 ± 0.06	0.14 ± 0.08
mnist09	0.45 ± 0.13	0.44 ± 0.12	0.48 ± 0.13
mnist14	0.47 ± 0.12	0.38 ± 0.07	0.51 ± 0.12
mnist27	0.32 ± 0.12	0.29 ± 0.10	0.35 ± 0.13
mnist38	0.22 ± 0.10	0.20 ± 0.08	0.24 ± 0.10
mnist56	0.30 ± 0.12	0.29 ± 0.11	0.32 ± 0.13
mushrooms	0.26 ± 0.06	0.26 ± 0.05	0.28 ± 0.07
musklarge	0.18 ± 0.09	0.15 ± 0.06	0.18 ± 0.09
ringnorm	0.15 ± 0.07	0.14 ± 0.06	0.15 ± 0.07
spambase	0.21 ± 0.13	0.19 ± 0.10	0.23 ± 0.13
splice	0.19 ± 0.12	0.18 ± 0.10	0.22 ± 0.14
twonorm	0.29 ± 0.14	0.26 ± 0.11	0.30 ± 0.14
w4a	0.27 ± 0.11	0.23 ± 0.07	0.38 ± 0.12
waveform	0.25 ± 0.17	0.22 ± 0.14	0.28 ± 0.19
wine	0.13 ± 0.15	0.13 ± 0.14	0.12 ± 0.14
wisconsin	0.39 ± 0.15	0.35 ± 0.12	0.59 ± 0.21

Table 6.2: Mean and standard deviation of margins.

slightly smaller margins compared to AdaBoost but with much smaller variances. However, both EBBoost and ABR show accuracy improvements over AdaBoost. It is believed that the improvements of ABR are due to its ability to handle noisy situations and outliers more gracefully. The performance advantage of EBBoost is justified by the empirical Bernstein bound (the initial motivation). Typically, the margin distribution bounds do not explicitly account for variance information; an interesting direction for future research is to explore the relationship between the empirical Bernstein bounds as well as previous analyses of the margin distribution.

6.5 Summary

A novel boosting algorithm based on the empirical Bernstein inequality was proposed. The algorithm is as easy to implement as AdaBoost and is as efficient computationally (it does not require an expensive line search). EBBoost showed significant empirical advantages over AdaBoost. This chapter demonstrates that it is possible to design efficient algorithms based on sample variance penalization and to obtain improvements in test error. This chapter essentially showed the advantage of trading-off between the loss and the variance (for an exponential loss).

Chapter 7

Conclusions

This thesis proposed a generic idea called the "relative margin" to overcome certain limitations associated with large margin solutions. The key contributions of this thesis are the following:

- A sensitivity of large margin methods to affine transformations was identified.
- Several possible solutions were discussed and a particularly easy to implement and efficient formulation called the relative margin machines was proposed.
- Relative margin machine was shown to outperform the support vector machine on several datasets (both synthetic and real world).
- Generalization bounds for the proposed algorithms were derived using the notion of Rademacher complexity. This derivation involved a novel method of landmark examples to overcome certain difficulties associated with a data-dependent function class.
- The idea of relative margin is not restricted just to binary classification problems. Two useful extensions–to structured prediction and graph Laplacian spectrum learning–were shown.
- Based on the recent bounds involving empirical variance, an efficient boosting algorithm was proposed as well.

There are several possible future directions to pursue this line of research. One obvious direction of research is that nearly any large margin learning algorithm can be extended to a large relative margin learning. Maximum margin matrix factorization [Srebro *et al.*, 2005; Rennie and Srebro, 2005], maximum entropy discrimination [Jaakkola *et al.*, 1999] , multi-task learning [Evgeniou and Pontil, 2004] multiple kernel learning [Lanckriet *et al.*, 2004], semi-supervised learning [Belkin *et al.*, 2006] etc. are just some of the possible candidates for a relative margin extension.

The bounding constraints in the relative margin machine are unsupervised, meaning that they don't really require the labels. One natural extension is to exploit unlabeled examples in bounding constraints in a semi-supervised (or a transductive setup).

The approaches proposed in this thesis considered the first and second order moments. A natural question is: how do the higher order moments contribute to learning? Are there efficient algorithms to exploit higher order moments?

Appendix A

Appendix

A.1 McDiarmid's inequality

Assume $X_1, X_2, \ldots, X_n$ are independent random variables from a set $\mathcal{X}$ and $g : \mathcal{X}^n \to \mathbb{R}$. If the function g satisfies $\sup_{X_1,\ldots,X_n,\hat{X}_k} |g(X_1,\ldots,X_n) - g(X_1,\ldots,\hat{X}_k,\ldots,X_n)| \leq c_k$, for all $1 \leq k \leq n$ then, for any $\epsilon > 0$:

$$\mathbf{P}\left[g(X_1,\ldots,X_n) - \mathbf{E}[g(X_1,\ldots,X_n)] \geq \epsilon\right] \leq \exp\left(-\frac{2\epsilon^2}{\sum_{i=1}^n c_i^2}\right), \tag{A.1}$$

$$\mathbf{P}\left[\mathbf{E}[g(X_1,\ldots,X_n)] - g(X_1,\ldots,X_n) \geq \epsilon\right] \leq \exp\left(-\frac{2\epsilon^2}{\sum_{i=1}^n c_i^2}\right),$$

where the expectations are over the random draws of $X_1, \ldots, X_n$. Here the constants $c_1, c_2, \cdots, c_n$ are called Lipschitz constants.

A.2 Lipschitz constants for Section 3.4

Lemma 32 *The upper bound on $\hat{R}(\mathcal{G}_{B,D}^U)$, namely $T_1(\mathbf{U},\mathbf{S})$, admits the Lipschitz constant:*

$$\frac{2\sqrt{2B}}{\bar{D}n}\left(\sqrt{\sum_{i=1}^n \mathbf{x}_i^\top \mathbf{x}_i} - \sqrt{\sum_{i=1}^n \mathbf{x}_i^\top \mathbf{x}_i - \frac{DR^2\mu_{max}}{n\bar{D}+DR^2}}\right).$$

Proof The quantity of interest is the worst change in

$$\frac{2\sqrt{2B}}{n}\sqrt{\sum_{i=1}^n \mathbf{x}_i(\bar{D}\mathbf{I} + \frac{D}{n}\sum_{j=1}^n \mathbf{u}_j\mathbf{u}_j^\top)^{-1}\mathbf{x}_i^\top}$$

when $\mathbf{u}_k$ is varied for any setting of $\mathbf{u}_1, \ldots, \mathbf{u}_{k-1}, \mathbf{u}_{k+1}, \ldots, \mathbf{u}_n$. Since $\sum_{j=1, j\neq k}^{n} \mathbf{u}_j \mathbf{u}_j^\top$ is positive semi-definite and inside the inverse operator, $\mathbf{u}_k$ will have the most extreme effect on the expression when $\sum_{j=1, j\neq k}^{n} \mathbf{u}_j \mathbf{u}_j^\top = \mathbf{0}$. Thus, consider:

$$\frac{2\sqrt{2B}}{n} \sqrt{\sum_{i=1}^{n} \mathbf{x}_i^\top \left(\bar{D}\mathbf{I} + \frac{D}{n} \mathbf{u}_k \mathbf{u}_k^\top \right)^{-1} \mathbf{x}_i}.$$

Apply the Woodbury matrix inversion identity to the term inside the square root:

$$\begin{aligned} \sum_{i=1}^{n} \mathbf{x}_i^\top \left(\bar{D}\mathbf{I} + \frac{D}{n} \mathbf{u}_k \mathbf{u}_k^\top \right)^{-1} \mathbf{x}_i &= \frac{1}{\bar{D}} \sum_{i=1}^{n} \mathbf{x}_i^\top \left(\mathbf{I} - \frac{\mathbf{u}_k \mathbf{u}_k^\top}{\frac{n\bar{D}}{D} + \mathbf{u}_k^\top \mathbf{u}_k} \right) \mathbf{x}_i \\ &= \frac{1}{\bar{D}} \left(\sum_{i=1}^{n} \mathbf{x}_i^\top \mathbf{x}_i - \frac{\sum_{i=1}^{n} (\mathbf{x}_i^\top \mathbf{u}_k)^2}{\frac{n\bar{D}}{D} + \mathbf{u}_k^\top \mathbf{u}_k} \right). \end{aligned}$$

The maximum value of this expression occurs when $\mathbf{u}_k = \mathbf{0}$. To find the minimum, write the second term inside the brackets in the above expression as below:

$$\left(\frac{\mathbf{u}_k^\top}{\|\mathbf{u}_k\|} \sum_{i=1}^{n} \mathbf{x}_i \mathbf{x}_i^\top \frac{\mathbf{u}_k}{\|\mathbf{u}_k\|} \right) / \left(\frac{n\bar{D}}{D\mathbf{u}_k^\top \mathbf{u}_k} + 1 \right).$$

Clearly, in the numerator, the magnitude of $\mathbf{u}_k$ does not matter. To maximize this expression, $\mathbf{u}_k$ should be set to a vector of maximal length and in the same direction as the maximum eigenvector of $\sum_{i=1}^{n} \mathbf{x}_i \mathbf{x}_i^\top$. Since all examples are assumed to have bounded norm no larger than R, the largest $\mathbf{u}_k$ vector has norm R. Denoting the maximum eigenvalue of $\sum_{i=1}^{n} \mathbf{x}_i \mathbf{x}_i^\top$ by μ_{max}, it is easy to show the claimed value of Lipschitz constant for any k. ■

Lemma 33 *The upper bound on $\hat{R}(\mathcal{H}_{E,D}^{V})$, namely $T_2(\mathbf{V}, \mathbf{S})$, admits the Lipschitz constant:*

$$\frac{2\sqrt{2E}}{\bar{D}n} \left(\sqrt{\sum_{i=1}^{n} \mathbf{x}_i^\top \mathbf{x}_i} - \sqrt{\sum_{i=1}^{n} \mathbf{x}_i^\top \mathbf{x}_i - \frac{DR^2 \mu_{max}}{n\bar{D} + DR^2}} \right).$$

Proof The quantity of interest is the maximum change in the following optimization problem over $\mathbf{u}_k$ for any setting of $\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_{k-1}, \mathbf{u}_{k+1}, \ldots, \mathbf{u}_{n_v}$:

$$\min_{\lambda \geq 0} \; \frac{1}{n} \sum_{i=1}^{n} \mathbf{x}_i^\top \left(\bar{D} \sum_{j=1}^{n_v} \lambda_j \mathbf{I} + D \sum_{j=1}^{n_v} \lambda_j \mathbf{u}_j \mathbf{u}_j^\top \right)^{-1} \mathbf{x}_i + \frac{2}{n} E \sum_{i=1}^{n_v} \lambda_i.$$

As before, this happens when all $\mathbf{u}$'s except $\mathbf{u}_k$ are $\mathbf{0}$. In such a scenario, the expression is minimized for the setting $\lambda_j = 0$ for all $j \neq k$. The minimization only needs to consider variable settings of λ_k. Since this minimization is over a single scalar, it is possible to obtain a closed-form expression for λ_k. The optimal λ_k is merely: $\frac{1}{\sqrt{2E}} \sum_{i=1}^{n} \mathbf{x}_i^\top \left(\bar{D}\mathbf{I} + D\mathbf{u}_k\mathbf{u}_k^\top\right)^{-1} \mathbf{x}_i$. Substituting this into the objective gives an expression which is independent of λ's:

$$\frac{2\sqrt{2E}}{n} \sqrt{\sum_{i=1}^{n} \mathbf{x}_i^\top \left(\bar{D}\mathbf{I} + \frac{D}{n}\mathbf{u}_k\mathbf{u}_k^\top\right)^{-1} \mathbf{x}_i}.$$

This expression is identical to the one obtained in Theorem 32 and the proof follows. ■

A.3 Solving for n_v

Let $x = \frac{1}{\sqrt{n_v}}, c = 4R\sqrt{\frac{2E}{D}}$ and $b = \frac{3}{2}\sqrt{\frac{\ln(2/\delta)}{2}}$. Consider solving for x in the expression $x^2 - 2bx = (c+2b)/\sqrt{n}$. Equivalently, solve $(x-b)^2 = b^2 + (c+2b)/\sqrt{n}$. Taking the square root of both sides gives $x = b \pm \sqrt{b^2 + (c+2b)/\sqrt{n}}$. Since $x > 0$, only the positive root is considered. Thus, $\sqrt{n_v} = 1/(b + \sqrt{b^2 + (c+2b)/\sqrt{n}})$ which gives an exact expression for n_v. Dropping terms from the denominator produces the simpler expression: $\sqrt{n_v} \leq 1/\sqrt{(c+2b)/\sqrt{n}}$. Hence, $n_v \leq \frac{\sqrt{n}}{4R\sqrt{\frac{2E}{D}} + 3\sqrt{\frac{\ln(2/\delta)}{2}}}$.

Bibliography

[Altun *et al.*, 2003a] Y. Altun, T. Hofmann, and M. Johnson. Discriminative learning for label sequence via boosting. In *In Advances in Neural Information Processing Systems 15*, 2003.

[Altun *et al.*, 2003b] Y. Altun, I. Tsochantaridis, and T. Hofmann. Hidden markov support vector machines. In *In Proceedings of the Twentieth International Conference on Machine Learning*, 2003.

[Asuncion and Newman, 2007] A. Asuncion and D.J. Newman. UCI machine learning repository, 2007.

[Audibert *et al.*, 2007] Jean-Yves Audibert, Rémi Munos, and Csaba Szepesvári. Tuning bandit algorithms in stochastic environments. In *18th Algorithmic Learning Theory*, pages 150–165, 2007.

[Bach *et al.*, 2004] F. R. Bach, G. R. G. Lanckriet, and M. I. Jordan. Multiple kernel learning, conic duality, and the smo algorithm. In *International Conference on Machine Learning*, 2004.

[Bakir *et al.*, 2006] G. Bakir, T. Hofmann, B. Schölkopf, A. J. Smola, B. Taskar, and S.V.N. Vishwanathan, editors. *Predicting Structured Data*. MIT Press, Cambridge, MA, 2006.

[Bartlett and Mendelson, 2002] P. L. Bartlett and S. Mendelson. Rademacher and Gaussian complexities: Risk bounds and structural results. *Journal of Machine Learning Research*, 3:463–482, 2002.

[Bartlett *et al.*, 2006] P. L. Bartlett, M. I. Jordan, and J. D. McAuliffe. Convexity, classification, and risk bounds. *Journal of the American Statistical Association*, 101(473):138–156, 2006.

[Belkin *et al.*, 2005] M. Belkin, P. Niyogi, and V. Sindhwani. On manifold regularization. In *Artificial Intelligence and Statistics*, 2005.

[Belkin *et al.*, 2006] M. Belkin, P. Niyogi, and V. Sindhwani. Manifold regularization: A geometric framework for learning from labeled and unlabeled examples. *Journal of Machine Learning Research*, 7:2399–2434, 2006.

[Bengio *et al.*, 2007] Y. Bengio, P. Lamblin, D. Popovici, and H. Larochelle. Greedy layer-wise training of deep networks. In B. Schölkopf, J. Platt, and T. Hoffman, editors, *Advances in Neural Information Processing Systems 19*, pages 153–160. MIT Press, Cambridge, MA, 2007.

[Bousquet *et al.*, 2004] O. Bousquet, S. Boucheron, and G. Lugosi. *Introduction to Statistical Learning Theory*, volume Lecture Notes in Artificial Intelligence 3176, pages 169–207. Springer, Heidelberg, Germany, 2004.

[Boyd and Vandenberghe, 2003] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2003.

[Burges, 1998] C. Burges. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2(2):121–167, 1998.

[Cawley and Talbot, 2003] G. C. Cawley and N. L. C Talbot. Efficient leave-one-out cross-validation of kernel fisher discriminant classifiers. *Pattern Recognition*, 36:2585–2592, 2003.

[Cesa-Bianchi *et al.*, 2002] N. Cesa-Bianchi, A. Conconi, and C. Gentile. A second-order perceptron algorithm. In *in Proc. 5th Annu. Conf. Computational Learning Theory (Lecture Notes in Artificial Intelligence). Berlin, Germany*, volume 2375, pages 121–137, 2002.

[Cesa-Bianchi *et al.*, 2005] N. Cesa-Bianchi, A. Conconi, and C. Gentile. A second-order perceptron algorithm. *SIAM Journal of Computing*, 34(3):640–668, 2005.

[Chow and Schwartz, 1991] Y-L Chow and R Schwartz. The n-best algorithm: an efficient procedure for finding top n sentence hypotheses. In *In Proceedings of the IEEE International Conference on Acoustics, Speech and Signal processing*, pages 81–84, 1991.

[Crammer *et al.*, 2009a] K. Crammer, M. Dredze, and F. Pereira. Exact convex confidence-weighted learning. In *Advances in Neural Information Processing Systems 21*, Cambridge, MA, 2009. MIT Press.

[Crammer *et al.*, 2009b] K. Crammer, M. Mohri, and F. Pereira. Gaussian margin machines. In *Proceedings of the Artificial Intelligence and Statistics*, 2009.

[Cristianini *et al.*, 2001] N. Cristianini, J. Shawe-Taylor, A. Elisseeff, and J. S. Kandola. On kernel-target alignment. In *NIPS*, pages 367–373, 2001.

[Decoste and Schölkopf, 2002] D. Decoste and B. Schölkopf. Training invariant support vector machines. *Machine Learning*, pages 161–190, 2002.

[Dredze *et al.*, 2008] M. Dredze, K. Crammer, and F. Pereira. Confidence-weighted linear classification. In *Internationcal Conference on Machine Leaarning*, 2008.

[Duda *et al.*, 2000] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification*. Wiley-Interscience Publication, 2000.

[Evgeniou and Pontil, 2004] T. Evgeniou and M. Pontil. Regularized multi–task learning. In *KDD*, pages 109–117, 2004.

[Freund and Schapire, 1997] Y. Freund and R. E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119–139, 1997.

[Haffner, 2001] P. Haffner. Escaping the convex hull with extrapolated vector machines. In T. G. Dietterich, S. Becker, and Z. Ghahramani, editors, *Advances in Neural Information Processing Systems 14*, pages 753–760. MIT Press, Cambridge, MA, 2001.

[Hastie *et al.*, 2001] Trevor Hastie, Robert Tibshirani, and J. H. Friedman. *The elements of statistical learning: data mining, inference, and prediction*. New York: Springer-Verlag, 2001.

[Herbrich *et al.*, 2001] R. Herbrich, T. Graepel, and C. Campbell. Bayes point machines. *Journal of Machine Learning Research*, 1:245–279, 2001.

[Jaakkola *et al.*, 1999] T. Jaakkola, M. Meila, and T. Jebara. Maximum entropy discrimination. In *Advances in Neural Information Processing Systems 11*, 1999.

[Joachims, 1998] T. Joachims. Making large-scale support vector machine learning practical. In A. Smola B. Schölkopf, C. Burges, editor, *Advances in Kernel Methods: Support Vector Machines*. MIT Press, Cambridge, MA, 1998.

[Joachims, 1999] T. Joachims. Transductive inference for text classification using support vector machines. In *Proceedings of the 16th International Conference on Machine Learning*, 1999.

[Joachims, 2003] T. Joachims. Transductive learning via spectral graph partitioning. In *ICML*, pages 290–297, 2003.

[Joachims, 2006] T. Joachims. Training linear SVMs in linear time. In *ACM SIGKDD International Conference On Knowledge Discovery and Data Mining (KDD)*, pages 217–226, 2006.

[Keerthi, 2002] S.S. Keerthi. Efficient tuning of svm hyperparameters using radius/margin bound and iterative algorithms. *IEEE Transactions on Neural Networks*, 13:1225–1229, 2002.

[Koltchinskii and Panchenko, 2002] V. Koltchinskii and D. Panchenko. Empirical margin distributions and bounding the generalization error of combined classifiers. *Annals of Statistics*, 30, 2002.

[Kondor and Lafferty, 2002] R. I. Kondor and J. D. Lafferty. Diffusion kernels on graphs and other discrete input spaces. In *ICML*, pages 315–322, 2002.

[Lafferty *et al.*, 2001] J. Lafferty, A. McCallum, and F. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *In Proceedings of the Eighteenth International Conference on Machine Learning*, pages 282–289, 2001.

[Lanckriet *et al.*, 2004] G. R. G. Lanckriet, N. Cristianini, P. L. Bartlett, L. E. Ghaoui, and M. I. Jordan. Learning the kernel matrix with semidefinite programming. *Journal of Machine Learning Research*, 5:27–72, 2004.

[LeCun *et al.*, 1989] Y. LeCun, B. Boser, J.S. Denker, D. Henderson, R.E. Howard, W. Hubbard, and L. Jackel. Back-propagation applied to handwritten zip code recognition. *Neural Computation*, 1:541–551, 1989.

[LeCun *et al.*, 1998] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[Ma *et al.*, 2010] J. Ma, A. Kulesza, M. Dredze, K. Crammer, L. Saul, and F. Pereira. Exploiting feature covariance in high-dimensional online learning. In *Proceedings of the Artificial Intelligence and Statistics*, 2010.

[Maurer and Pontil, 2009] Andreas Maurer and Massimiliano Pontil. Empirical Bernstein bounds and sample variance penalization. In *22nd Annual Conference on Learning Theory*, 2009.

[Mika *et al.*, 1999] S. Mika, G. Raetsch, J. Weston, B. Scholkopf, and K.-R. Muller. Fisher discriminant analysis with kernels. In *in Neural Networks for Signal Processing IX*, pages 41–48, 1999.

[Mnih *et al.*, 2008] Volodymyr Mnih, Csaba Szepesvári, and Jean-Yves Audibert. Empirical Bernstein stopping. In *Proceedings of the Twenty-Fifth International Conference on Machine Learning*, pages 672–679, 2008.

[Raetsch *et al.*, 2001] G. Raetsch, T. Onoda, and K.-R. Muller. Soft margins for adaboost. *Machine Learning*, 43:287–320, 2001.

[Rennie and Srebro, 2005] J. D. M. Rennie and N. Srebro. Fast maximum margin matrix factorization for collaborative prediction. In *In Proceedings of the 22nd International Conference on Machine Learning (ICML*, pages 713–719. ACM, 2005.

[Schapire *et al.*, 1998] R. E. Schapire, Y. Freund, P. Bartlett, and W. S. Lee. Boosting the margin: a new explanation for the effectiveness of voting methods. *The Annals of Statistics*, 26:322–330, 1998.

[Schölkopf and Smola, 2002] B. Schölkopf and A.J. Smola. *Learning with Kernels*. MIT Press, Cambridge, MA, USA, 2002.

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

[Shalev-Shwartz *et al.*, 2007] S. Shalev-Shwartz, Y. Singer, and N. Srebro. Pegasos: Primal estimated sub-gradient solver for svm. In *International Conference on Machine Learning*, 2007.

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

[Shivaswamy and Jebara, 2007] P. K. Shivaswamy and T. Jebara. Ellipsoidal kernel machines. In *Proceedings of the Artificial Intelligence and Statistics*, 2007.

[Shivaswamy and Jebara, 2009] P. K. Shivaswamy and T. Jebara. Structured prediction with relative margin. In *International Conference on Machine Learning and Applications*, 2009.

[Shivaswamy and Jebara, 2010a] P. Shivaswamy and T. Jebara. Laplacian spectrum learning. In *European Conference on Machine Learning*, 2010.

[Shivaswamy and Jebara, 2010b] P. Shivaswamy and T. Jebara. Maximum relative margin and data-dependent regularization. *Journal of Machine Learning Research*, 11:747–788, 2010.

[Sinz *et al.*, 2008] F. Sinz, O. Chapelle, A. Agarwal, and B. Schölkopf. An analysis of inference with the universum. In J.C. Platt, D. Koller, Y. Singer, and S. Roweis, editors, *Advances in Neural Information Processing Systems 20*, pages 1369–1376. MIT Press, Cambridge, MA, 2008.

[Smola and Kondor, 2003] A. J. Smola and R. I. Kondor. Kernels and regularization on graphs. In *COLT*, pages 144–158, 2003.

[Srebro *et al.*, 2005] N. Srebro, J. D. M. Rennie, and T. S. Jaakola. Maximum-margin matrix factorization. In *Advances in Neural Information Processing Systems 17*, pages 1329–1336. MIT Press, 2005.

[Taskar *et al.*, 2004] B. Taskar, C. Guestrin, and D. Koller. Max-margin markov networks. *In Advances in Neural Information Processing Systems 16*, 2004.

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

[Tsochantaridis *et al.*, 2005] I. Tsochantaridis, T. Joachims, T. Hofmann, and Y. Altun. Large margin methods for structured and interdependent output variables. *Journal of Machine Learning Research (JMLR)*, 6:1453–1484, September 2005.

[Vapnik and Vashist, 2009] V. Vapnik and A. Vashist. A new learning paradigm: Learning using privileged information. *Neural Networks*, 22(5-6):544–557, 2009.

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

[Weston *et al.*, 2000] J. Weston, S. Mukherjee, O. Chapelle, M. Pontil, T. Poggio, and V. Vapnik. Feature selection for SVMs. In T. K. Leen, T. G. Dietterich, and V. Tresp, editors, *Advances in Neural Information Processing Systems 13*, pages 668–674. MIT Press, Cambridge, MA, 2000.

[Weston *et al.*, 2006] J. Weston, R. Collobert, F. H. Sinz, L. Bottou, and V. Vapnik. Inference with the universum. In *Proceedings of the International Conference on Machine Learning*, pages 1009–1016, 2006.

[Zhang *et al.*, 2005] B. Zhang, X. Chen, S. Shan, and W. Gao. Nonlinear face recognition based on maximum average margin criterion. In *Computer Vision and Pattern Recognition*, pages 554–559, 2005.

[Zhu *et al.*, 2003] X. Zhu, J. Lafferty, and Z. Ghahramani. Semi-supervised learning: From gaussian fields to gaussian processes. Technical report, Carnegie Mellon University, 2003.

[Zhu *et al.*, 2004] X. Zhu, J. S. Kandola, Z. Ghahramani, and J. D. Lafferty. Nonparametric transforms of graph kernels for semi-supervised learning. In *NIPS*, 2004.