Modern Classification with Big Data

A DISSERTATION
SUBMITTED TO THE FACULTY OF THE GRADUATE SCHOOL
OF THE UNIVERSITY OF MINNESOTA
BY

Boxiang Wang

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

Hui Zou, Adviser

July 2018

ACKNOWLEDGEMENTS

First of all, I would like to express my deepest gratitude to my adviser, Professor Hui Zou, who cared so much about my work with his invaluable guidance and great patience. I have benefited from his deep insights in statistics and his constant pursuit of high quality research with large impact in science and industry. I owe an enormous debt to him for his tremendous support and encouragement in my job application. He is not only the greatest adviser that I can imagine, but also a friend and life mentor. I could never thank him enough.

I would like to extend my gratitude to all the faculty and staff in School of Statistics. I thank Professor Galin Jones, Adam Rothman, Haitao Chu, Lan Wang, Vipin Kumar, for serving on my dissertation and defense committees. I would also like to thank Professor Dennis Cook and Yuhong Yang for their sound advice and generous help. I am also indebted to all my statistics teachers prior to my Ph.D. career. Special thanks go to Professor Zhaojun Wang, Minqian Liu, and Changliang Zou at Nankai University and Professor Arthur Yeh, Kenneth Ryan, Herb McGrath, Jane Chang, and John Chen in Bowling Green.

I have been very fortunate to work with all the members in Professor Zou's research group. I am especially grateful to my academic brothers Yi Yang, Yuwen Gu, and Abhirup Datta for their immense help both within and outside statistics. I also thank Professor Lingchen Kong for his guidance in optimization and thank Jun Fan and Di He for many inspiring discussions. I owe a debt to Lingzhou Xue and Qing Mai for their great help in my job application. A big thank you goes out to my fellow graduate students. Thank you, Zhuoran, Ding, Kevin, Yang, Yiwen, Yanjia, Kaibo, Si, Chenglong, Ming, and Lin. I have truly been blessed to have such an delightful and colorful Ford Fourth Floor life.

I am most grateful for the unconditional love and support from my family. Last but foremost, I thank my wife Fei Huang. I owe you much more than I would ever be able to express here. Thank you for always being there in every circumstance and making me the luckiest man in the world. Together with you, I can't wait to welcome our unborn baby.

DEDICATION

To the loving memory of my father, Yuzhong Wang.

ABSTRACT

Rapid advances in information technologies have ushered in the era of "big data" and revolutionized the scientific research across many disciplines, including economics, genomics, neuroscience, and modern commerce. Big data creates golden opportunities but has also arisen unprecedented challenges due to the massive size and complex structure of the data. Among many tasks in statistics and machine learning, classification has diverse applications, ranging from improving daily life to reaching the new frontiers of science and engineering. This thesis will discuss the envisions of broader approaches to modern classification methodologies, as well as computational considerations to cope with the big data challenges.

Chapter 2 of the thesis presents a modern classification method named data-driven generalized distance weighted discrimination. A fast algorithm with an emphasis on computational efficiency for big data will be introduced. Our method is formulated in a reproducing kernel Hilbert space, and learning theory of the Bayes risk consistency will be developed. We will use extensive benchmark data applications to demonstrate that the prediction accuracy of our method is highly competitive with state-of-the-art classification methods including support vector machine, random forest, gradient boosting, and deep neural network. Chapter 3 introduces sparse penalized DWD for high-dimensional classification, which is commonly used in the era of big data. We develop a very efficient algorithm to compute the solution path of the sparse DWD at a given fine grid of regularization parameters. Chapter 4 proposes multicategory kernel distance weighted discrimination for multi-class classification. The proposal is defined as a margin-vector optimization problem in a reproducing kernel Hilbert space. This formulation is shown to enjoy Fisher consistency. We develop an accelerated projected gradient descent algorithm to fit multicategory kernel DWD. Chapter 5 develops a magic formula for doing CV in the context of large margin classification. We design a novel and successful algorithm to fit and tune the support vector machine.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

From the days of *The Lady Tasting Tea* to the era of big data, statistical classification has diverse applications, ranging from improving daily life to reaching the new frontiers of science and engineering. These examples include spam e-mail filtering, cancer diagnosis, speech recognition, and image detection, to name a few. Nevertheless, many applications are bottlenecked by the demand for high classification accuracy and by the resulting huge computational cost; at the present stage, the existing classification methodologies can hardly meet those needs. Therefore, the research goal in this thesis is to take one step forward in expanding the reach and arch of statistics in the big data era, improving and innovating modern classification methodologies, as well as developing fast algorithms to reduce computational costs.

This thesis has a specific focus on the support vector machine (SVM, Vapnik, 1995, 1998), as it is unquestionably one of the best modern classification methods and it delivers exceptionally high prediction accuracy due to its large-margin geometry. The success of SVM is mainly attributed to its nice large-margin geometric interpretation and thereby called large-margin classifiers. The SVM produces the largest "margin" between the separating hyperplane and the closest data points, leading to great classification accuracy. Fernández-Delgado et al. (2014) showed that the classification accuracy of SVM is among the best of the 179 classifiers. To amplify the competitiveness and popularity of SVM, this thesis focuses on improving the SVM in two directions, *harmonic-margin generalization* and *model-selection integration*.

The *harmonic-margin generalization* of SVM stemmed from Marron et al. (2007), who

argued that some small-scale noise artifacts of the data may largely affect SVM and impair the generalization performance, especially when the dimension of the data is high. As a remedy, Marron et al. (2007) developed a new method termed distance weighted discrimination (DWD) by amending the margin definition in SVM, and they presented many numerical examples to demonstrate the superior performance of DWD over SVM. In Chapter 2 of this thesis, we find that DWD actually inaugurates a harmonic-margin framework, which is indexed by q and treats the SVM as a limiting case. In this regard, DWD is definitely worth more effort and we thus advanced the entire harmonic-margin framework of DWD as a modern classifier, orienting it for the big data era.

In Chapter 2, we make DWD computationally amiable for big data. The DWD was originally solved based on second-order-cone programming (SOCP), which was computationally demanding and only works for the special case of $q = 1$ under the harmonic-margin framework. We further propose a surprisingly simple algorithm based on the majorization-minimization principle, which is unified for any q. We show that our algorithm can be thousands of times faster than the SOCP implementation, and it efficiently handles data with millions of observations.

Furthermore, prior to this work, only linear DWD was available and how to formulate kernel DWD and establish Bayes risk consistency had been a long-lasting and open research question. In Chapter 2, we formulate kernel DWD in reproducing kernel Hilbert spaces, and we strictly prove that the prediction error of kernel DWD asymptotically converges to the Bayes error in theory. To prove this, we decompose the difference between the Bayes error and the kernel DWD prediction error into approximation error and estimation error; we first prove the approximation error is zero when employing a universal kernel such as Gaussian kernel, on the basis of Lusin's theorem in functional analysis. We then illustrate the estimation error of kernel DWD diminishes as the sample size increases. The Bayes risk consistency theory is favored by extensive numerical studies on benchmark data with various sample sizes and dimensions. We show that the classification accuracy of kernel DWD is no worse than any other modern classification method, such as random forest and deep neural net. Our method has been implemented in an R package `kerndwd`, which is on CRAN and has received more than 10,000 downloads.

In Chapter 3, we push DWD to high-dimensionality, which is an important compo-

nent of big data and frequently arises in domains like DNA microarray technology. For high-dimensional data analysis, variable selection plays pivotal roles in knowledge discovery and guarding the error accumulation from noise variables. In Chapter 3, we develop sparse penalized DWD with several penalties for high-dimensional classification. Besides the methodology, as high-dimensional classification is computationally more challenging, we derive an efficient algorithm hinged on coordinate-descent. We have published the implementation of our algorithm as an R package `sdwd` on CRAN (7859 downloads, as of June 30, 2018).

In Chapter 4, we advocate DWD for multi-category classification. We develop a multi-category kernel DWD in reproducing kernel Hilbert spaces. We adopt the concept *margin vector* which can be regarded as a proxy of conditional class probabilities. Computationally, we develop a very efficient accelerated projected gradient descent algorithm. Theoretically, we prove that our formulation has the multi-category Fisher consistency. As shown in Chapter 4, the prediction accuracy of our method is highly competitive with other state-of-the-art multi-category classifiers.

In Chapter 5, we investigate the *model-selection integration* of the SVM. Cross-validation (CV) is perhaps the most widely used technique for tuning supervised machine learning algorithms. We first advocate the leave-one-out cross-validation (LOOCV) in estimating the generalization error: it is well known that LOOCV has almost no bias, while we also show that LOOCV has variance no larger than any other V-fold cross-validation. This in fact addresses a seemingly widespread misconception that LOOCV has the largest variance.

On the other hand, LOOCV has remarkably large computational cost due to the repetitions in the model fits. The leave-one-out lemma (Craven and Wahba, 1979) was proposed to offer a closed-form LOOCV estimates. For classification, nevertheless, all existing fast cross-validation attempts were either complicated, narrowly applicable, or unable to yield exact cross-validation estimates. In Chapter 5, we develop a magic formula for doing CV in the context of large margin classification. Based on the magic CV formula, we propose a magic SVM, which is a novel algorithm to integrate the processes of fitting and tuning the SVM. The algorithm is entirely different from the existing SVM algorithm. We show that, with yielding almost identical results, the magic SVM is significant faster than the state-of-the-art kernel SVM solvers such as `kernlab` and `e1071`.

Chapter 2

Distance Weighted Discrimination

Distance weighted discrimination (DWD) is a modern margin-based classifier with an interesting geometric motivation. It was proposed as a competitor to the support vector machine (SVM). Despite many recent references on DWD, DWD is far less popular than the SVM, mainly because of computational and theoretical reasons. In this chapter, we greatly advance the current DWD methodology and its learning theory. We propose a novel thrifty algorithm for solving standard DWD and generalized DWD, and our algorithm can be several hundred times faster than the existing algorithm based on the second-order cone programming. In addition, we exploit the new algorithm to design an efficient scheme to tune generalized DWD. Furthermore, we formulate a natural kernel DWD approach in a reproducing kernel Hilbert space and then establish the Bayes risk consistency of the kernel DWD by using a universal kernel such as the Gaussian kernel. This result solves an open theoretical problem in the DWD literature. A comparison study on 16 benchmark data sets shows that data-driven generalized DWD consistently delivers higher classification accuracy with less computation time than the SVM.

2.1 Introduction

Binary classification problems appear from diverse practical applications, such as, financial fraud detection, spam e-mail classification, medical diagnosis with genomics data, drug response modeling, among many others. In these classification problems, the goal is to predict class labels based on a given set of variables. Suppose that we observe a training data

set consisting of n pairs, where $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$, $\boldsymbol{x}_i \in \mathbb{R}^p$, and $y_i \in \{-1, 1\}$. A classifier fits a discriminant function f and constructs a classification rule to classify data point $\boldsymbol{x_i}$ to either class 1 or class -1 according to the sign of $f(\boldsymbol{x}_i)$. The decision boundary is given by $\{\boldsymbol{x} : f(\boldsymbol{x}) = 0\}$. Two canonical classifiers are linear discriminant analysis and logistic regression. Modern classification algorithms can produce flexible non-linear decision boundaries with high accuracy. The two most popular approaches are ensemble learning and support vector machines or kernel machines. Ensemble learning such as boosting (Freund and Schapire, 1997) and random forest (Breiman, 2001) combine many weak learners like decision trees into a powerful one. The support vector machine (SVM) (Vapnik, 1995, 1998) fits an optimal separating hyperplane in the extended kernel feature space which is non-linear in the original covariate spaces. In a recent extensive numerical study by Fernández-Delgado et al. (2014), the kernel SVM was shown to be one of the best among 179 commonly used classifiers.

Marron et al. (2007) invented a new classification algorithm named distance weighted discrimination (DWD), which retains the elegant geometric interpretation of the SVM, resolves a "data-piling" issue, and reveals competitive performance. Since then much work has been devoted to the development of DWD. Readers are referred to Marron (2015) for an up-to-date list of work on DWD. However, DWD is still only known to a small group of researchers. We can think of two reasons for that. First, DWD has algorithmic disadvantages compared with the SVM. The current algorithm for DWD is based on second-order cone programming (SOCP). As acknowledged in Marron et al. (2007), SOCP was then (and still is) much less well-known than quadratic programming. In addition, SOCP is generally more computationally demanding than quadratic programming. Second, the kernel extension of DWD and the corresponding kernel learning theory are underdeveloped compared to the kernel SVM. Although Marron et al. (2007) proposed a version of non-linear DWD by mimicking the kernel trick used for deriving the kernel SVM, theoretical justification of such a kernel DWD is still absent. In the constrast, the kernel SVM as well as kernel logistic regression (Wahba et al., 1994; Zou and Hastie, 2005) have mature theoretical understandings built on the theory of reproducing kernel Hilbert spaces (RKHS) (Wahba, 1999; Hastie et al., 2009). How to establish the Bayes risk consistency of kernel DWD was proposed as a fundamental open problem in the original DWD paper (Marron et al., 2007).

A decade later, the problem still remains open.

In this chapter, we aim to resolve the aforementioned issues. We show that the kernel DWD in an RKHS has the Bayes risk consistency property if a universal kernel is used. This result solves the open problem in DWD literature. We also develop a novel fast algorithm to solve the linear and kernel DWD by using the majorization-minimization (MM) principle. Our new algorithm also easily handles the generalized DWD. To summarize, our new algorithm has multiple advantages over the SOCP algorithm: it is much faster to compute, easier to understand, and capable of solving the generalized DWD and conducting efficient tuning. We have implemented our algorithms in an R package `kerndwd`, which is publicly available on CRAN at http://cran.r-project.org/web/packages/kerndwd/index.html.

The rest of the chapter is organized as follows. To be self-contained, we first review the SVM and DWD in Section 2.2. We then derive a novel algorithm for DWD and generalized DWD in Section 2.3. We introduce the kernel DWD in a reproducing kernel Hilbert space and establish its Bayes risk consistency in Section 2.4. Numeric examples are given in Section 2.5, and technical proofs are provided in Appendix B.

2.2 Review of the SVM and DWD

In this section we give a brief review of the SVM and DWD. Hastie et al. (2009) offered a highly detailed discussion of the SVM and its kernel version. Marron (2015) provided a more comprehensive review of the current DWD literature.

Both the SVM and DWD share a common geometric interpretation. Consider a case when two classes are separable by a hyperplane $\{\boldsymbol{x} : f(x) = \omega_0 + \boldsymbol{x}^T\boldsymbol{\omega} = 0\}$ where $y_i(\omega_0 + \boldsymbol{x}_i^T\boldsymbol{\omega})$ are all non-negative. Without loss of generality, we assume that $\boldsymbol{\omega}$ is a unit vector, i.e., $\boldsymbol{\omega}^T\boldsymbol{\omega} = 1$, and we observe that each $d_i \equiv y_i(\omega_0 + \boldsymbol{x}_i^T\boldsymbol{\omega})$ is equivalent to the Euclidean distance between the data point $\boldsymbol{x}_i$ and the hyperplane. The reason is that $d_i = (\boldsymbol{x}_i - \boldsymbol{x}_0)^T\boldsymbol{\omega}$ and $\omega_0 + \boldsymbol{x}_0^T\boldsymbol{\omega} = 0$, where $\boldsymbol{x}_0$ is any data point on the hyperplane and $\boldsymbol{\omega}$ is the unit normal vector. The SVM classifier is defined as the optimal separating hyperplane that maximizes $\min_i d_i$, the smallest distance of each data point to the separating hyperplane (Vapnik, 1995). In a more general case when the two classes are not separable, non-negative

slack variables η_i are introduced to ensure that all $y_i(\omega_0 + \boldsymbol{x}_i^T\boldsymbol{\omega}) + \eta_i$ are non-negative. So, the SVM is defined as

$$\begin{aligned} \max_{\omega_0,\boldsymbol{\omega},d_i,\eta_i} \quad & \min d_i, \\ \text{subject to} \quad & d_i = y_i(\omega_0 + \boldsymbol{x}_i^T\boldsymbol{\omega}) + \eta_i \geq 0, \ \forall i, \\ & \eta_i \geq 0, \ \forall i, \ \sum_{i=1}^{n} \eta_i < \text{constant}, \ \text{and } \boldsymbol{\omega}^T\boldsymbol{\omega} = 1. \end{aligned} \tag{2.1}$$

The data points that are closest to the hyperplane, i.e., $d_i = \min d_i$, are dubbed the support vectors. The SVM can be solved by rephrasing problem (2.1) as a quadratic programming problem. A more general kernel SVM can be derived by applying the so-called kernel trick (Aizerman et al., 1964) on the dual formulation of the linear SVM (Hastie et al., 2009).

Marron et al. (2007) proposed DWD that finds a separating hyperplane minimizing the total inverse margins of all the data points:

$$\begin{aligned} \min_{\omega_0,\boldsymbol{\omega},d_i,\eta_i} \quad & \left[\sum_{i=1}^{n} \frac{1}{d_i} + c\sum_{i=1}^{n} \eta_i\right], \\ \text{subject to} \quad & d_i = y_i(\omega_0 + \boldsymbol{x}_i^T\boldsymbol{\omega}) + \eta_i \geq 0, \ \eta_i \geq 0, \ \forall i, \ \text{and } \boldsymbol{\omega}^T\boldsymbol{\omega} = 1. \end{aligned} \tag{2.2}$$

The motivation of Marron et al. (2007) is to have a method that is directly formulated by an SVM-like margin-maximization picture and also exhausts all data points for classification. In some sense, DWD is like a blend of the SVM and a more classical logistic regression.

Marron et al. (2007) solved DWD by reformulating problem (2.2) as a second-order cone programming (SOCP) program (Alizadeh and Goldfarb, 2004; Boyd and Vandenberghe, 2004), which has a linear objective, linear constraints, and second-order cone constraints. Specifically, for each i, let $\rho_i = (1/d_i + d_i)/2$, $\sigma_i = (1/d_i - d_i)/2$, and then $\rho_i + \sigma_i = 1/d_i$, $\rho_i - \sigma_i = d_i$, and $\rho_i^2 - \sigma_i^2 = 1$. Hence the original optimization problem

(2.2) becomes

$$\begin{aligned} \min_{\omega_0,\boldsymbol{\omega},\rho_i,\sigma_i,\eta_i} \quad & \left[\sum_{i=1}^{n}(\rho_i+\sigma_i)+c\sum_{i=1}^{n}\eta_i\right], \\ \text{subject to} \quad & \rho_i-\sigma_i=y_i\boldsymbol{x}_i^T\boldsymbol{\omega}+\omega_0\cdot y_i+\eta_i,\ \forall i, \\ & \eta_i\geq 0,\ (\rho_i;\sigma_i,1)\in S_3,\ \forall i,\ (1;\boldsymbol{\omega})\in S_{p+1}, \end{aligned} \tag{2.3}$$

in which $S_{m+1}=\{(\psi,\boldsymbol{\phi})\in\mathbb{R}^{m+1}:\psi^2\geq\boldsymbol{\phi}^T\boldsymbol{\phi}\}$ is the form of the second-order cones.

There has been much work on variants of the standard DWD. We can give only an incomplete list here. Qiao et al. (2010) introduced weighted DWD to tackle unequal cost or sample sizes by imposing different weights on two classes. Huang et al. (2013) extended the DWD to the multiclass case. We shall propose sparse DWD for high-dimensional classification in Chapter 3 of this thesis. In addition, the work connecting DWD with other classifiers, e.g., the SVM, includes but not limited to LUM (Liu et al., 2011), DWSVM (Qiao and Zhang , 2015a), and FLAME (Qiao and Zhang , 2015b).

Marron et al. (2007) also attempted to replace the reciprocal in the standard DWD optimization problem (2.2) with the qth power ($q>0$) of the inverse distances. Hall et al. (2005) also used it as the original definition of DWD. We name the DWD with this new formulation the generalized DWD:

$$\begin{aligned} \min_{\omega_0,\boldsymbol{\omega}} \quad & \left[\sum_{i=1}^{n}\frac{1}{d_i^q}+c\sum_{i=1}^{n}\eta_i\right], \\ \text{subject to} \quad & d_i=y_i(\omega_0+\boldsymbol{x}_i^T\boldsymbol{\omega})+\eta_i\geq 0,\ \eta_i\geq 0,\ \forall i,\ \text{and}\ \boldsymbol{\omega}^T\boldsymbol{\omega}=1, \end{aligned} \tag{2.4}$$

which degenerates to the standard DWD (2.2) when $q=1$. Generalized DWD has not been implemented yet because the SOCP transformation only works for the standard DWD ($q=1$) (2.2), but its extension to handle the general cases is unclear if not impossible. That is why the current DWD literature focuses only on DWD with $q=1$. In fact, generalized DWD with $q\neq 1$ was proposed as an open research problem in Marron et al. (2007). The new algorithm that is proposed in this chapter can easily solve the generalized DWD problem for any $q>0$; see Section 2.3.

Another open research problem that was proposed in Marron et al. (2007) is regarding

the Bayes risk consistency of kernel DWD. Marron et al. (2007) followed the similar kernel trick in the kernel SVM to design kernel DWD. In short, Marron et al. (2007) used the Cholesky decomposition of the kernel matrix, i.e., $\boldsymbol{K} = \boldsymbol{\Phi}\boldsymbol{\Phi}^T$ and then replaced the design matrix $\boldsymbol{X}$ with $\boldsymbol{\Phi}$. To our best knowledge, still unclear is the theoretical justification for the kernel DWD in Marron et al. (2007). The reason is likely to be that the nonlinear extension is purely algorithmic. Kernel DWD considered in this chapter can be rigorously justified to have a universal Bayes risk consistency property; see details in Section 2.4.2.

2.3 A Novel Algorithm for DWD and Generalized DWD

In this section we introduce a new algorithm that offers a unified efficient solution to standard DWD and generalized DWD.

2.3.1 Generalized DWD loss

Our algorithm begins with a $loss - plus - penalty$ formulation of the DWD. Lemma 2.1 deploys the result. Note that the loss function also lays the foundation of the kernel DWD learning theory that will be discussed in Section 2.4.

Lemma 2.1

The generalized DWD classifier in (2.4) can be written as $\text{sign}(\hat{\beta}_0 + \boldsymbol{x}_i^T\hat{\boldsymbol{\beta}})$, where $(\hat{\beta}_0, \hat{\boldsymbol{\beta}})$ is computed for some λ from

$$\min_{\beta_0,\boldsymbol{\beta}} \boldsymbol{C}(\beta_0, \boldsymbol{\beta}) \equiv \min_{\beta_0,\boldsymbol{\beta}} \left[\frac{1}{n} \sum_{i=1}^{n} V_q \left(y_i(\beta_0 + \boldsymbol{x}_i^T\boldsymbol{\beta}) \right) + \lambda \boldsymbol{\beta}^T\boldsymbol{\beta} \right]; \tag{2.5}$$

$$V_q(u) = \begin{cases} 1 - u, & \text{if } u \leq \dfrac{q}{q+1}, \\ \dfrac{1}{u^q} \dfrac{q^q}{(q+1)^{q+1}}, & \text{if } u > \dfrac{q}{q+1}. \end{cases} \tag{2.6}$$

□

By Lemma 2.1, we call $V_q(\cdot)$ the generalized DWD loss. When $q = 1$, the generalized DWD loss becomes

$$V_1(u) = \begin{cases} 1 - u, & \text{if } u \leq 1/2, \\ 1/(4u), & \text{if } u > 1/2. \end{cases}$$

We notice that $V_1(\cdot)$ has appeared in the literature (Qiao et al., 2010; Liu et al., 2011).

2.3.2 Derivation of the algorithm

We now show how to develop the new algorithm by using the MM principle (De Leeuw and Heiser, 1977; Lange et al., 2000; Hunter and Lange, 2004). Some recent successful applications of the MM principle can be seen in Hunter and Li (2005); Wu and Lange (2010); Zou and Li (2008); Zhou and Lange (2010); Yang and Zou (2013); Lange and Zhou (2014), among others. The main idea of the MM principle is easy to understand. Suppose $\boldsymbol{\theta} = (\beta_0, \boldsymbol{\beta}^T)^T$ and we aim to minimize $\boldsymbol{C}(\boldsymbol{\theta})$, defined in problem (2.5). The MM principle finds a majorization function $\boldsymbol{D}(\boldsymbol{\theta}|\boldsymbol{\theta}_k)$ satisfying $\boldsymbol{C}(\boldsymbol{\theta}) < \boldsymbol{D}(\boldsymbol{\theta}|\boldsymbol{\theta}_k)$ for any $\boldsymbol{\theta} \neq \boldsymbol{\theta}_k$ and $\boldsymbol{C}(\boldsymbol{\theta}_k) = \boldsymbol{D}(\boldsymbol{\theta}_k|\boldsymbol{\theta}_k)$, and then we generate a sequence $\{\boldsymbol{C}(\boldsymbol{\theta}_k)\}_{k=1}^{\infty}$ via $\boldsymbol{\theta}_{k+1} = \operatorname{argmin}_{\boldsymbol{\theta}} \boldsymbol{D}(\boldsymbol{\theta}|\boldsymbol{\theta}_k)$.

We first expose some properties of the generalized DWD loss function, which gives rise to a quadratic majorization function of $\boldsymbol{C}(\boldsymbol{\theta})$. The generalized DWD loss is differentiable everywhere; its first-order derivative is given by

$$V_q'(u) = \begin{cases} -1, & \text{if } u \leq \dfrac{q}{q+1}, \\ -\dfrac{1}{u^{q+1}} \left(\dfrac{q}{q+1} \right)^{q+1}, & \text{if } u > \dfrac{q}{q+1}. \end{cases}$$

Lemma 2.2

The generalized DWD loss function $V_q(\cdot)$ has a Lipschitz continuous gradient,

$$|V_q'(t) - V_q'(\tilde{t})| < \frac{(q+1)^2}{q} |t - \tilde{t}|, \tag{2.7}$$

which further implies a quadratic majorization function of $V_q(\cdot)$ such that for any $t \neq \tilde{t}$,

$$V_q(t) < V_q(\tilde{t}) + V_q'(\tilde{t})(t-\tilde{t}) + \frac{(q+1)^2}{2q}(t-\tilde{t})^2. \tag{2.8}$$

□

For a given pair of (q, λ), denote the current solution by $\tilde{\boldsymbol{\theta}} = (\tilde{\beta}_0, \tilde{\boldsymbol{\beta}}^T)^T$ and the updated solution by $\boldsymbol{\theta} = (\beta_0, \boldsymbol{\beta}^T)^T$. We set $\boldsymbol{C}(\boldsymbol{\theta}) = \boldsymbol{C}(\beta_0, \boldsymbol{\beta})$ and $\boldsymbol{D}(\boldsymbol{\theta}|\tilde{\boldsymbol{\theta}}) = \boldsymbol{D}(\beta_0, \boldsymbol{\beta})$ without abusing the notations. Let $\boldsymbol{X}$ be an $n \times p$ data matrix with the ith row $\boldsymbol{x}_i^T$, $\tilde{\boldsymbol{z}}$ be an $n \times 1$ vector with the ith element $y_i V_q'(y_i(\tilde{\beta}_0 + \boldsymbol{x}_i^T\tilde{\boldsymbol{\beta}}))/n$, and $\mathbf{1} \in \mathbb{R}^n$ be a vector of ones. We see that

$$\begin{aligned}
\boldsymbol{C}(\beta_0, \boldsymbol{\beta}) \equiv & \frac{1}{n}\sum_{i=1}^{n} V_q\left(y_i(\beta_0 + \boldsymbol{x}_i^T\boldsymbol{\beta})\right) + \lambda\boldsymbol{\beta}^T\boldsymbol{\beta} \\
\leq & \frac{1}{n}\sum_{i=1}^{n} V_q\left(y_i(\tilde{\beta}_0 + \boldsymbol{x}_i^T\tilde{\boldsymbol{\beta}})\right) + \lambda\tilde{\boldsymbol{\beta}}^T\tilde{\boldsymbol{\beta}} \\
& + \tilde{\boldsymbol{\gamma}}^T \begin{pmatrix} \beta_0 - \tilde{\beta}_0 \\ \boldsymbol{\beta} - \tilde{\boldsymbol{\beta}} \end{pmatrix} + \frac{(q+1)^2}{2nq} \begin{pmatrix} \beta_0 - \tilde{\beta}_0 \\ \boldsymbol{\beta} - \tilde{\boldsymbol{\beta}} \end{pmatrix}^T \boldsymbol{P}_{q,\lambda} \begin{pmatrix} \beta_0 - \tilde{\beta}_0 \\ \boldsymbol{\beta} - \tilde{\boldsymbol{\beta}} \end{pmatrix} \\
\equiv & \boldsymbol{D}(\beta_0, \boldsymbol{\beta}),
\end{aligned}$$

where

$$\tilde{\boldsymbol{\gamma}} \equiv \begin{pmatrix} \mathbf{1}^T\tilde{\boldsymbol{z}} \\ \boldsymbol{X}^T\tilde{\boldsymbol{z}} + 2\lambda\tilde{\boldsymbol{\beta}} \end{pmatrix}, \ \boldsymbol{P}_{q,\lambda} \equiv \begin{pmatrix} n & \mathbf{1}^T\boldsymbol{X} \\ \boldsymbol{X}^T\mathbf{1} & \boldsymbol{X}^T\boldsymbol{X} + \frac{2nq\lambda}{(q+1)^2}\boldsymbol{I}_{p\times p} \end{pmatrix},$$

and the inequality comes from Lemma 2.2 with the equality held only when $(\beta_0, \boldsymbol{\beta}) = (\tilde{\beta}_0, \tilde{\boldsymbol{\beta}})$. To minimize $\boldsymbol{D}(\beta_0, \boldsymbol{\beta})$, we set $[\partial\boldsymbol{D}(\beta_0, \boldsymbol{\beta})/\partial\beta_0, \partial\boldsymbol{D}(\beta_0, \boldsymbol{\beta})/\partial\boldsymbol{\beta}]$ to be zeros, and then we have

$$\begin{pmatrix} \beta_0 \\ \boldsymbol{\beta} \end{pmatrix} = \begin{pmatrix} \tilde{\beta}_0 \\ \tilde{\boldsymbol{\beta}} \end{pmatrix} - \frac{nq}{(q+1)^2}\boldsymbol{P}_{q,\lambda}^{-1}\tilde{\boldsymbol{\gamma}}. \tag{2.9}$$

2.3.3 Efficient tuning

When using DWD in practice, we often need to compute its solution for a grid of λ values and then use the data to find a good λ for the final DWD classifier. If we directly apply the MM algorithm that was discussed above, the matrix $\boldsymbol{P}_{q,\lambda}$ is repeatedly inverted as q and λ are varied. Inverting a large matrix can be costly. Here we exploit the special structure of $\boldsymbol{P}_{q,\lambda}$ such that we need to invert the matrix only one time for all candidate pairs of (q, λ).

For the sake of discussion we consider the usual $n > p$ case. The inversion of a $p \times p$ matrix costs $\mathcal{O}(p^3)$ operations. For the linear DWD with $n < p$, we can treat it as special kernel DWD with a linear kernel and then we only need to invert an $n \times n$ matrix. Our treatment also works for the kernel DWD, which will be discussed in Section 4.

We first define a matrix $\boldsymbol{P}_0$ and compute its eigendecomposition:

$$\boldsymbol{P}_0 \equiv \begin{pmatrix} n & \mathbf{1}^T\boldsymbol{X} \\ \boldsymbol{X}^T\mathbf{1} & \boldsymbol{X}^T\boldsymbol{X} \end{pmatrix} = \boldsymbol{U}\boldsymbol{\Pi}\boldsymbol{U}^T, \tag{2.10}$$

where $\boldsymbol{\Pi}$ is a diagonal matrix such that $(\boldsymbol{\Pi})_{ii} = d_i$, the ith eigenvalue of $\boldsymbol{P}_0$. For each q and λ, we craft a matrix $\boldsymbol{Q}_{q,\lambda}$,

$$\boldsymbol{Q}_{q,\lambda} \equiv \begin{pmatrix} n + \frac{2nq\lambda}{(q+1)^2} & \mathbf{1}^T\boldsymbol{X} \\ \boldsymbol{X}^T\mathbf{1} & \boldsymbol{X}^T\boldsymbol{X} + \frac{2nq\lambda}{(q+1)^2}\boldsymbol{I}_{p\times p} \end{pmatrix} = \boldsymbol{U}\boldsymbol{\Pi}_{q,\lambda}\boldsymbol{U}^T,$$

where $\boldsymbol{\Pi}_{q,\lambda}$ is a diagonal matrix whose ith diagonal element is $d_i + \frac{2nq\lambda}{(q+1)^2}$. We then disintegrate $\boldsymbol{P}_{q,\lambda}^{-1}$ by using the Sherman-Morrison formula:

$$\boldsymbol{P}_{q,\lambda}^{-1} = \left(\boldsymbol{Q}_{q,\lambda} + \begin{pmatrix} -\frac{2nq\lambda}{(q+1)^2} & \mathbf{0}^T \\ \mathbf{0} & \mathbf{0}_{p\times p} \end{pmatrix}\right)^{-1} = \boldsymbol{Q}_{q,\lambda}^{-1} + g\boldsymbol{v}\boldsymbol{v}^T = \boldsymbol{U}\boldsymbol{\Pi}_{q,\lambda}^{-1}\boldsymbol{U}^T + g\boldsymbol{v}\boldsymbol{v}^T, \tag{2.11}$$

where $\boldsymbol{v}$ is the first column of $\boldsymbol{Q}_{q,\lambda}^{-1}$ and $g = 2nq\lambda/[(q+1)^2 - 2nq\lambda\mathbf{1}^T\boldsymbol{v})]$.

Finally, we directly compute $\boldsymbol{P}_{q,\lambda}^{-1}\tilde{\boldsymbol{\gamma}}$ in equation (2.9) rather than $\boldsymbol{P}_{q,\lambda}^{-1}$. By equation

(2.11), we see that

$$\boldsymbol{P}_{q,\lambda}^{-1}\boldsymbol{\gamma} = (\boldsymbol{Q}_{q,\lambda}^{-1} + g\boldsymbol{v}\boldsymbol{v}^T)\tilde{\boldsymbol{\gamma}} = \boldsymbol{U}\boldsymbol{\Pi}_{q,\lambda}^{-1}\boldsymbol{U}^T\tilde{\boldsymbol{\gamma}} + g\boldsymbol{v}\boldsymbol{v}^T\tilde{\boldsymbol{\gamma}}. \tag{2.12}$$

Note that $\boldsymbol{v}$ can be obtained via $\boldsymbol{U}\boldsymbol{\Pi}_{q,\lambda}^{-1}\boldsymbol{u}_1$ where $\boldsymbol{u}_1$ is the first row of $\boldsymbol{U}$, hence we find that all the operations in equation (2.12) are $\mathcal{O}(p^2)$. Therefore, we manage to do the actual matrix inversion once in equation (2.10), when solving the generalized DWD with different q and λ values.

2.4 Kernel DWD in RKHS and Bayes Risk Consistency

2.4.1 Kernel DWD in RKHS

The kernel SVM can be derived by using the kernel trick or using the view of non-parametric function estimation in a reproducing kernel Hilbert space (RKHS). Much of the theoretical work on the kernel SVM is based on the RKHS formulation of SVMs. The derivation of the kernel SVM in an RKHS is given in Hastie et al. (2009). We take a similar approach to derive kernel DWD, as our goal is to establish kernel learning theory for DWD.

Consider $\mathcal{H}_K$, an RKHS generated by the kernel function K. Mercer's theorem ensures K to have an eigen-expansion $K(\boldsymbol{x}, \boldsymbol{x}') = \sum_{t=1}^{\infty} \gamma_t \phi_t(\boldsymbol{x})\phi_t^T(\boldsymbol{x}')$, with $\gamma_t \geq 0$ and $\sum_{t=1}^{\infty} \gamma_t^2 < \infty$. Then the Hilbert space $\mathcal{H}_K$ is defined as the collection of functions $h(\boldsymbol{x}) = \sum_{t=1}^{\infty} \theta_t \phi_t(\boldsymbol{x})$, for any θ_t such that $\sum_{t=1}^{\infty} \theta_t^2/\gamma_t < \infty$, and the inner product is

$$\left\langle \sum_{t=1}^{\infty} \theta_t \phi_t(\boldsymbol{x}), \sum_{t'=1}^{\infty} \delta_{t'} \phi_{t'}(\boldsymbol{x}) \right\rangle_{\mathcal{H}_K} = \sum_{t=1}^{\infty} \theta_t \delta_t / \gamma_t.$$

Given $\mathcal{H}_K$, let non-linear DWD be $\text{sign}(\hat{\beta}_0 + \hat{h}(\boldsymbol{x}))$ where $(\hat{\beta}_0, \hat{h})$ is the solution of

$$\min_{\substack{h \in \mathcal{H}_K \\ \beta_0 \in \mathbb{R}}} \left[\frac{1}{n} \sum_{i=1}^{n} V_q \left(y_i(\beta_0 + h(\boldsymbol{x}_i)) \right) + \lambda ||h||_{\mathcal{H}_K}^2 \right], \tag{2.13}$$

where $V_q(\cdot)$ is the generalized DWD loss (2.6). The representer theorem concludes that

the solution of problem (2.13) has a finite expansion based on $K(\boldsymbol{x}, \boldsymbol{x}_i)$ (Wahba, 1990), $\hat{h}(\boldsymbol{x}) = \sum_{i=1}^{n} \hat{\alpha}_i K(\boldsymbol{x}, \boldsymbol{x}_i)$, and thus $||\hat{h}||^2_{\mathcal{H}_K} = \sum_{i=1}^{n}\sum_{j=1}^{n} \hat{\alpha}_i\hat{\alpha}_j K(\boldsymbol{x}_i, \boldsymbol{x}_j)$. Consequently, problem (2.13) can be paraphrased with matrix notation,

$$\min_{\beta_0,\boldsymbol{\alpha}} \boldsymbol{C}_K(\beta_0, \boldsymbol{\alpha}) \equiv \min_{\beta_0,\boldsymbol{\alpha}} \left[\frac{1}{n}\sum_{i=1}^{n} V_q\left(y_i(\beta_0 + \boldsymbol{K}_i^T\boldsymbol{\alpha})\right) + \lambda\boldsymbol{\alpha}^T\boldsymbol{K}\boldsymbol{\alpha}\right],$$

where $\boldsymbol{K}$ is the kernel matrix with $(\boldsymbol{K})_{i,j} = K(\boldsymbol{x}_i, \boldsymbol{x}_j)$ and $\boldsymbol{K}_i$ is the ith column of $\boldsymbol{K}$.

The procedure for deriving the linear DWD algorithm can be directly adopted for solving the kernel DWD. Define $\tilde{\boldsymbol{z}} = (z_1, \ldots, z_n)^T$ with each $z_i = y_i V_q'(y_i(\tilde{\beta}_0 + \boldsymbol{K}_i\tilde{\boldsymbol{\alpha}}))/n$, and

$$\boldsymbol{P}_{q,\lambda} \equiv \begin{pmatrix} n & \boldsymbol{1}^T\boldsymbol{K} \\ \boldsymbol{K}\boldsymbol{1} & \boldsymbol{K}\boldsymbol{K} + \frac{2nq\lambda}{(q+1)^2}\boldsymbol{K} \end{pmatrix}.$$

We define the majorization function $\boldsymbol{D}_K(\beta_0, \boldsymbol{\alpha}) \geq \boldsymbol{C}_K(\beta_0, \boldsymbol{\alpha})$, where the equality holds only when $(\beta_0, \boldsymbol{\alpha}) = (\tilde{\beta}_0, \tilde{\boldsymbol{\alpha}})$:

$$\begin{aligned}\boldsymbol{D}_K(\beta_0, \boldsymbol{\alpha}) =& \frac{1}{n}\sum_{i=1}^{n} V_q\left(y_i(\tilde{\beta}_0 + \boldsymbol{K}_i^T\tilde{\boldsymbol{\alpha}})\right) + \lambda\tilde{\boldsymbol{\alpha}}^T\boldsymbol{K}\tilde{\boldsymbol{\alpha}} \\ &+ \begin{pmatrix} \boldsymbol{1}^T\tilde{\boldsymbol{z}} \\ \boldsymbol{K}\tilde{\boldsymbol{z}} + 2\lambda\boldsymbol{K}\tilde{\boldsymbol{\alpha}} \end{pmatrix}^T \begin{pmatrix} \beta_0 - \tilde{\beta}_0 \\ \boldsymbol{\alpha} - \tilde{\boldsymbol{\alpha}} \end{pmatrix} + \frac{(q+1)^2}{2nq}\begin{pmatrix} \beta_0 - \tilde{\beta}_0 \\ \boldsymbol{\alpha} - \tilde{\boldsymbol{\alpha}} \end{pmatrix}^T \boldsymbol{P}_{q,\lambda}\begin{pmatrix} \beta_0 - \tilde{\beta}_0 \\ \boldsymbol{\alpha} - \tilde{\boldsymbol{\alpha}} \end{pmatrix},\end{aligned}$$

whose close-form minimizer is obtained as

$$\begin{pmatrix} \beta_0 \\ \boldsymbol{\alpha} \end{pmatrix} = \begin{pmatrix} \tilde{\beta}_0 \\ \tilde{\boldsymbol{\alpha}} \end{pmatrix} - \frac{nq}{(q+1)^2}\boldsymbol{P}_{q,\lambda}^{-1}\begin{pmatrix} \boldsymbol{1}^T\boldsymbol{z} \\ \boldsymbol{K}\boldsymbol{z} + 2\lambda\boldsymbol{K}\tilde{\boldsymbol{\alpha}} \end{pmatrix}.$$

The direct use of this formula requires computing the inverse of $n \times n$ matrix $\boldsymbol{P}_{q,\lambda}$ repeatedly for different values of q and λ. We find a way to do the matrix inversion that costs $\mathcal{O}(n^3)$ operations only once. We first compute the eigendecomposition $\boldsymbol{K} = \boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{U}^T$, which is free of tuning parameters, and we then compute $\boldsymbol{\Pi}_{q,\lambda} = \boldsymbol{\Lambda}\boldsymbol{\Lambda} + \frac{2nq\lambda}{(q+1)^2}\boldsymbol{\Lambda}$ for each q and λ. With $\boldsymbol{v} = \boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{\Pi}_{q,\lambda}^{-1}\boldsymbol{U}^T\boldsymbol{1}$ being computed through $\mathcal{O}(n^2)$ operations and $g =$

$1/(n - \mathbf{1}^T\boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{\Pi}_{q,\lambda}^{-1}\boldsymbol{\Lambda}\boldsymbol{U}^T\mathbf{1})$, we glean the decomposition:

$$\boldsymbol{P}_{q,\lambda}^{-1} = \begin{pmatrix} n & \mathbf{1}^T\boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{U}^T \\ \boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{U}^T\mathbf{1} & \boldsymbol{U}\boldsymbol{\Pi}_{q,\lambda}\boldsymbol{U}^T \end{pmatrix}^{-1} = g \begin{pmatrix} 1 \\ -\boldsymbol{v} \end{pmatrix} \begin{pmatrix} 1 & -\boldsymbol{v}^T \end{pmatrix} + \begin{pmatrix} 0 & \mathbf{0}^T \\ \mathbf{0} & \boldsymbol{U}\boldsymbol{\Pi}_{\lambda}^{-1}\boldsymbol{U}^T \end{pmatrix}.$$

Rather than compute $\boldsymbol{P}_{q,\lambda}^{-1}$, we direct compute

$$\boldsymbol{P}_{q,\lambda}^{-1} \begin{pmatrix} \mathbf{1}^T\boldsymbol{z} \\ \boldsymbol{K}\boldsymbol{z} + 2\lambda\boldsymbol{K}\tilde{\boldsymbol{\alpha}}_q \end{pmatrix} = g\left(\mathbf{1}^T\boldsymbol{z} - \boldsymbol{v}^T\boldsymbol{K}(\boldsymbol{z} + 2\lambda\tilde{\boldsymbol{\alpha}}_q)\right) \begin{pmatrix} 1 \\ -\boldsymbol{v} \end{pmatrix} + \begin{pmatrix} 0 \\ \boldsymbol{U}\boldsymbol{\Pi}_{q,\lambda}^{-1}\boldsymbol{\Lambda}\boldsymbol{U}^T(\boldsymbol{z} + 2\lambda\tilde{\boldsymbol{\alpha}}_q) \end{pmatrix},$$

where only $\mathcal{O}(n^2)$ operations abound. The computation time is appreciably reduced.

2.4.2 Kernel learning theory

Lin (2002) formulated the kernel SVM as a non-parametric function estimation problem in an RKHS and showed that the population minimizer of the SVM loss function is the Bayes rule, indicating that the SVM directly approximates the optimal Bayes classifier. Vapnik-Chervonenkis (VC) analysis (Vapnik, 1998; Anthony and Bartlett, 1999) and margin analysis (Bartlett and Shawe-Taylor, 1999; Shawe-Taylor and Cristianini, 2000) have been used to bound the expected classification error of the SVM. Zhang (2004) used so-called leave-one-out analysis (Jaakkola and Haussler, 1999) to study a class of kernel machines. The existing theoretical work on the kernel SVM provides us a nice road map to study kernel DWD. In this section we first elucidate the Fisher consistency (Lin, 2004) of the generalized kernel DWD, and we then establish the Bayes risk consistency of kernel DWD when a universal kernel is employed.

Let $\eta(\boldsymbol{x})$ denote the conditional probability $P(Y = 1|\boldsymbol{X} = \boldsymbol{x})$. Under 0-1 loss, the theoretical optimal Bayes rule is $f^\star(\boldsymbol{x}) = \text{sign}(\eta(\boldsymbol{x}) - 1/2)$. Assume $\eta(\boldsymbol{x})$ is a measurable function and $P(\eta(\boldsymbol{x}) = 1/2) = 0$ throughout.

Lemma 2.3

The population minimizer of the expected DWD loss $E_{\boldsymbol{X}Y}[V_q\left(Yf(\boldsymbol{X})\right)]$ is

$$\tilde{f}(\boldsymbol{x}) = \frac{q}{q+1}\left[\left(\frac{\eta(\boldsymbol{x})}{1-\eta(\boldsymbol{x})}\right)^{\frac{1}{q+1}} \cdot I(\eta(\boldsymbol{x}) > 1/2) - \left(\frac{1-\eta(\boldsymbol{x})}{\eta(\boldsymbol{x})}\right)^{\frac{1}{q+1}} \cdot I(\eta(\boldsymbol{x}) < 1/2)\right]. \tag{2.14}$$

□

The population minimizer $\tilde{f}(\boldsymbol{x})$ has the same sign as $\eta(\boldsymbol{x}) - 1/2$.

Lin (2004) coined the name "Fisher consistency" to explain why a margin-based loss function is appropriate for fitting a classifier, besides its convexity property for computational considerations. Following Lin (2004), we see from Lemma 2.3 that the generalized DWD loss is Fisher consistent. Lemma 2.3 is a generalization of a previously shown result (Qiao et al., 2010; Liu et al., 2011) that proves the standard DWD loss $V_1(u)$ is Fisher consistent. It is worth emphasizing that the condition and conclusion of Lemma 2.3 are independent of the functional space: Fisher consistency only hinges on the loss function. Lemma 2.3 is treated as an important intermediate step in our theoretical analysis.

In reality all classifiers are estimated from a finite sample. Thus, a more refined analysis of the actual DWD classifier is needed, and that is what we achieve in follows. Such results are missing in the current DWD literature.

Following the convention in the literature, we absorb the intercept into h and present the kernel DWD as follows:

$$\hat{f}_n = \underset{f\in\mathcal{H}_K}{\operatorname{argmin}}\left[\frac{1}{n}\sum_{i=1}^{n} V_q\left(y_i(f(\boldsymbol{x}_i)) + \lambda_n||f||^2_{\mathcal{H}_K}\right.\right]. \tag{2.15}$$

The ultimate goal is to show that the misclassification error of the kernel DWD approaches the Bayes error rate such that we can say that kernel DWD classifier works as well as the Bayes rule (asymptotically speaking). For this, we present Lemma 2.4. The lemma is closely related to Theorem 2.1 of Zhang (2004) and Theorem 3 of Bartlett et al. (2006).

Lemma 2.4

For a discrimination function f, we define $R(f) = E_{\boldsymbol{X}Y}\left[Y \neq \text{sign}\left(f(\boldsymbol{X})\right)\right]$. Assume that $f^\star = \text{argmin}_f\, R(f)$ is the Bayes rule and $\hat{f}_n$ is the solution of (2.15), then

$$R(\hat{f}_n) - R(f^\star) \leq \frac{q+1}{q}(\varepsilon_A + \varepsilon_E), \tag{2.16}$$

where ε_A and ε_E are defined as follows and V_q is the generalized DWD loss,

$$\varepsilon_A = \inf_{f \in \mathcal{H}_K} E_{\boldsymbol{X}Y}\left[V_q(Yf(\boldsymbol{X}))\right] - E_{\boldsymbol{X}Y}\left[V_q\left(Y\tilde{f}(\boldsymbol{X})\right)\right],$$
$$\varepsilon_E = \varepsilon_E(\hat{f}_n) = E_{\boldsymbol{X}Y}\left[V_q\left(Y\hat{f}_n(\boldsymbol{X})\right)\right] - \inf_{f \in \mathcal{H}_K} E_{\boldsymbol{X}Y}\left[V_q(Yf(\boldsymbol{X}))\right].$$

□

In the above lemma $R(f^*)$ is the Bayes error rate and $R(\hat{f}_n)$ is the mis-classification rate of kernel DWD applied to new data points. If $R(\hat{f}_n) \to R(f^\star)$, we say the classifier is Bayes risk consistent. On the basis of Lemma 2.4, it suffices to show that both ε_A and ε_E approach 0 in order to demonstrate the Bayes risk consistency of kernel DWD. Note that ε_A is deterministic and is called the approximation error. If the RKHS is rich enough then the approximation error can be made arbitrarily small. In the literature, the notation of *universal kernel* (Steinwart, 2001; Micchelli et al., 2006) has been proposed and studied. Suppose $\mathcal{X} \in \mathbb{R}^p$ is the compact input space of $\boldsymbol{X}$ and $C(\mathcal{X})$ is the space of all continuous functions $g : \mathcal{X} \to \mathbb{R}$. The kernel K is said to be *universal* if the function space $\mathcal{H}_K$ generated by K is dense in $\mathcal{C}(\mathcal{X})$, that is, for any positive ϵ and any function $g \in \mathcal{C}(\mathcal{X})$, there exists an $f \in \mathcal{H}_K$ such that $||f - g||_\infty < \epsilon$.

Theorem 2.1

Suppose $\hat{f}_n$ is the solution of (2.15), $\mathcal{H}_K$ is induced by a universal kernel K, and the sample space $\mathcal{X}$ is compact. Then we have

(1) $\varepsilon_A = 0$.

(2) Let $B = \sup_{\boldsymbol{x}} K(\boldsymbol{x}, \boldsymbol{x}) < \infty$. When $\lambda_n \to 0$ and $n\lambda_n \to \infty$, for any $\epsilon > 0$,

$$\lim_{n\to\infty} P\left(\varepsilon_E(\hat{f}_n) > \epsilon\right) = 0.$$

By (1) and (2) and (2.16) we have $R(\hat{f}_n) \to R(f^*)$ in probability.

The Gaussian kernel is universal and $B \leq 1$. Thus Theorem 2.1 says kernel DWD using the Gaussian kernel is Bayes risk consistent. A proof of Theorem 2.1 is given in the Appendix B.

2.5 Numeric Examples

2.5.1 Timing comparison with SOCP implementations

We first use a few simulation models to demonstrate the superior computation performance of `kerndwd` over the R package `DWD` (Huang et al., 2012) and the Matlab software (Marron, 2013). Since the R package DWD and the Matlab implementation only solve linear DWD with $q = 1$, we only report the timing of computing the linear DWD with $q = 1$ in this set of simulations. The simulation models were designed by Marron et al. (2007). We adopted these models but changed the model dimension size to $n = 500$ and $p = 50$. In Table 2.1, we use `WZpath` to represent the time of using `kerndwd` to compute a solution path at 100 λ-values. We also denote by `WZ`, `HUANG`, and `MARRON` the time of computing DWD at the best λ using the R package `kerndwd`, `DWD`, and the Matlab implementation respectively.

From Table 2.1 we observe that both `WZpath` and `WZ` are much faster than `HUANG` and `MARRON`. In all four examples, the computation time of `WZ` was above 1000 times faster than `HUANG`, and also more than 100 times faster than `MARRON`. We also checked the quality of the computed solutions by these different algorithms. In theory they should be identical. In practice, because of machine errors and implementations, they could be different. We found that in all examples our new algorithm gave better solutions in the sense that the objective function in problem (2.3) has the smallest value. `HUANG` and `MARRON` gave similar but slightly larger objective function values.

Table 2.1: Computation time comparisons among the R package `kerndwd`, the R package `DWD` (denoted as `HUANG`), and the Matlab implementation (`MARRON`). We use `WZpath` to represent the computation time of using `kerndwd` to compute a solution path of DWD on 100 λ values, and `WZ` to represent the time of fitting DWD on the best λ. Both `HUANG` and `MARRON` only solve the DWD on the best λ. In four examples, the sample size $n = 500$ and the dimension $p = 50$. All the time is averaged over 100 independent replicates. Computations were conducted on a single processor Intel(R) Xeon(R) CPU E5-2660@2.60GHz.

	Average Time (in seconds)				Ratio	
	`WZpath`	`WZ`	`HUANG`	`MARRON`	$\frac{t(\texttt{HUANG})}{t(\texttt{WZ})}$	$\frac{t(\texttt{MARRON})}{t(\texttt{WZ})}$
1	0.206	0.008	12.646	1.065	1580.7	133.2
2	0.188	0.011	12.477	1.135	1171.5	106.6
3	0.199	0.009	11.963	1.009	1391.0	117.3
4	0.123	0.008	12.257	1.051	1602.2	137.4

2.5.2 Timing comparison with sGS-ADMM algorithm

A referee pointed out that a new method for solving DWD has been proposed in Lam et al. (2017) in which they devised a three-block inexact symmetric Gauss-Seidel-based semi-proximal ADMM, which is modified from very advanced mathematical programming work Li et al. (2016) and Chen et al. (2017), for computing DWD on large-scale data sets. Their new algorithm is implemented in a Matlab toolbox called `DWDLarge`. Table 2.2 summarizes the timing comparisons between `DWDLarge` and `kerndwd` on 10 benchmark data sets obtained from UCI Machine Learning Repository (Dua, D. and Karra Taniskidou, E., 2017). Among the ten data: `heart` and `ionosphere` have moderate n and p, `colon` and `leuk` have $n \ll p$, `gisette` has both large n and p, and the other five data sets have $n \gg p$. Notably, `covtype` has the sample size over a half million. `DWDLarge` only implements linear DWD with $q = 1, 2, 3, 4$.

In Table 2.2, we use `WZpath` to represent the run time of `kerndwd` when yielding the solution paths at 100 λ-values, and we use `WZ` and `LMST` to denote the time of solving DWD at the best λ, using `kerndwd` and `DWDLarge`, respectively. From Table 2.2 we observe that the timings of `WZpath` are only several times those of `WZ`, which indicates that our algorithm is efficient in computing the entire solution path. We also discover the

Table 2.2: Computation time of `kerndwd` and `DWDLarge` on ten UCI benchmark data sets. `WZpath` represents the run time of using `kerndwd` to compute the solution path of DWD at 100 λ values. `WZ` and `LMST` represent the run time of computing DWD at the best λ using `kerndwd` and `DWDLarge`, respectively. All the computation time is averaged over 100 independent replicates. Computations were conducted on a single processor Intel(R) Xeon(R) CPU E5-2660@2.60GHz.

data	n	p	q	WZpath	WZ	LMST	q	WZpath	WZ	LMST
heart	270	13	**1**	0.009	0.002	0.172	**3**	0.008	0.002	0.032
			2	0.008	0.002	0.036	**4**	0.009	0.002	0.053
ionosphere	351	33	**1**	0.071	0.017	0.179	**3**	0.049	0.013	0.050
			2	0.060	0.012	0.062	**4**	0.046	0.013	0.124
colon	62	2000	**1**	0.013	0.011	1.483	**3**	0.012	0.009	1.629
			2	0.012	0.010	1.204	**4**	0.011	0.010	2.989
leuk	72	7128	**1**	0.061	0.027	0.873	**3**	0.027	0.023	0.505
			2	0.027	0.023	1.012	**4**	0.026	0.023	0.825
a8a	22696	123	**1**	5.977	2.376	1.152	**3**	4.471	1.822	4.048
			2	4.936	1.956	1.839	**4**	4.175	1.756	23.592
a9a	32561	123	**1**	10.258	4.373	1.602	**3**	7.655	3.467	6.704
			2	8.541	3.718	2.995	**4**	5.425	2.395	23.768
ijcnn1	35000	22	**1**	4.045	1.602	2.730	**3**	2.567	1.026	5.638
			2	3.089	1.219	2.696	**4**	2.238	0.897	42.069
skin	245057	3	**1**	4.841	1.801	5.058	**3**	3.948	1.053	56.987
			2	4.150	1.072	19.529	**4**	3.875	1.179	286.073
covtype	581012	54	**1**	78.585	15.929	73.941	**3**	66.068	13.265	838.485
			2	63.566	11.338	76.538	**4**	68.391	13.679	751.353
gisette	6000	5000	**1**	2615.847	500.180	89.286	**3**	1284.623	420.025	86.394
			2	1608.612	428.477	132.788	**4**	1153.301	421.652	318.190

computation times of `WZpath` and `WZ` are quite consistent over various q values, whereas `LMST` becomes much slower when q is large. When the data have moderate n and p, or $n \ll p$, both `WZpath` and `WZ` are faster than `LMST`. In the case when $n \gg p$, we discover our methods are roughly on the same order of magnitude as `LMST` but `LMST` is time-consuming when $q = 4$. When both n and p are large, we find that `LMST` is more efficient than our algorithm.

2.5.3 Comparing the SVM and the generalized DWD

We now compare DWD and SVM in terms of classification accuracy and computation speed. We generated data from a mixture Gaussian model with dimension $p = 300$ and sample size n ranging from 100 to 1300. Define $\boldsymbol{\mu}_+ = (1, \ldots, 1, 0, \ldots, 0)$ and $\boldsymbol{\mu}_- = (0, \ldots, 0, 1, \ldots, 1)$, both of which consist of 150 ones and 150 zeros. In each example, the positive class arose from a mixture Gaussian distribution $\sum_{k=1}^{10} 0.1N(\boldsymbol{\mu}_{k+}, 10^2\boldsymbol{I})$ with each $\boldsymbol{\mu}_{k+}$ drawn from $N(\boldsymbol{\mu}_+, \boldsymbol{I})$, and likewise the negative class was assembled by $\sum_{k=1}^{10} 0.1N(\boldsymbol{\mu}_{k-}, 10^2\boldsymbol{I})$ with each $\boldsymbol{\mu}_{k-}$ from $N(\boldsymbol{\mu}_-, \boldsymbol{I})$. For this model the Bayes rule is nonlinear and the Bayes error is 18.85%. Using both linear and Gaussian kernels, we trained and tuned the SVM, DWD with fixed q from a wide range $\{0.01, 1, 10, 10^5\}$, and DWD with a data-driven q embracing a two-dimensional cross-validation of the pair (q, λ). We computed all the DWD methods by using our R package `kerndwd` and solved the SVM by using the R package `kernlab` (Karatzoglou et al., 2004). Similar results of the SVM were obtained by the R package `e1071` (Meyer et al., 2015).

Table 2.3 summarizes the average time for training the classifier and the mis-classification rates assessed on an independent test set with sample size of $10,000$. We can make several observations from Table 2.3. First, none of the kernel DWD with a fixed q dominates the others in prediction accuracy. Second, DWD with a data-driven q closely follows the best classifier, and it consistently outperforms the SVM. Third, as the sample size increases, the mis-classification rates of SVM and all variants of DWD are approaching the Bayes error rate. Fourth, in terms of computation time, all variants of DWD are much faster than the SVM. For example, when $n = 900$, even DWD with a data-driven q was about 65 times faster than the SVM in the linear case, and it was about 10 times faster in the kernel case.

We remark that the main algorithm for solving the SVM in the `kernlab` is through the sequential minimal optimization (Platt , 1999), which is entirely different from the MM algorithm that was used for DWD. Hence the difference that is revealed in Table 2.3 is mainly due to the different methods rather than the different implementations.

Many kernel methods including the SVM and DWD have difficulty in dealing with huge data sets. From Table 2.3 it is clear that the kernel SVM took a very long time to fit when n is 1300. DWD equipped wth our algorithm can handle the large sample size better

Table 2.3: Mis-classification rates and computation time, averaged by 20 runs, under mixture Gaussian distributed data. In each example, we used a five-fold cross-validation to tune the SVM, DWD with fixed q, and DWD with a data-driven q. We investigated the classification error on independently generated test sets and exhibited the standard errors in parentheses. We displayed all the time that includes fitting models and tuning the parameters with five-fold cross-validations. In each example, the method incurring the lowest error is marked by a black box.

	Bayes: 18.85	$n = 100, p = 300$			$n = 500, p = 300$			$n = 900, p = 300$			$n = 1300, p = 300$		
		error (%)		time	error (%)		time	error (%)		time	error (%)		time
Linear kernel	SVM	38.43	(0.00)	12.07	27.43	(0.09)	92.99	25.74	(0.38)	346.24	23.18	(0.09)	15325.31
	$\text{DWD}_{q=0.01}$	38.35	(0.00)	0.11	25.03	(0.22)	1.50	22.72	(0.09)	2.21	21.27	(0.05)	2.53
	$\text{DWD}_{q=1}$	35.43	(0.39)	0.09	21.97	(0.04)	2.12	20.05	(0.06)	2.79	19.00	(0.02)	2.87
	$\text{DWD}_{q=10}$	35.79	(0.37)	0.12	22.17	(0.07)	1.66	20.40	(0.15)	2.15	19.55	(0.05)	2.49
	$\text{DWD}_{q=10^5}$	39.07	(0.00)	0.11	28.49	(0.00)	1.53	24.92	(0.00)	2.25	22.36	(0.00)	2.35
	$\text{DWD}_{\text{data-driven } q}$	35.75	(0.66)	0.18	22.17	(0.35)	3.93	20.40	(0.13)	5.13	19.55	(0.15)	5.95
Gaussian kernel	SVM	35.97	(0.07)	13.48	23.00	(0.23)	138.52	20.50	(0.17)	587.38	19.42	(0.11)	1094.80
	$\text{DWD}_{q=0.01}$	35.19	(0.07)	0.55	**21.83**	(0.08)	6.20	20.18	(0.05)	22.27	**18.88**	(0.04)	45.53
	$\text{DWD}_{q=1}$	35.27	(0.07)	0.35	22.07	(0.05)	6.85	19.98	(0.09)	26.02	18.92	(0.05)	41.88
	$\text{DWD}_{q=10}$	35.37	(0.10)	0.64	22.09	(0.05)	7.57	20.00	(0.05)	28.80	19.01	(0.02)	53.17
	$\text{DWD}_{q=10^5}$	**35.16**	(0.26)	0.50	22.05	(0.29)	4.63	**19.86**	(0.31)	18.43	18.97	(0.33)	36.69
	$\text{DWD}_{\text{data-driven } q}$	35.35	(0.12)	0.83	21.94	(0.12)	13.48	20.07	(0.11)	48.97	19.00	(0.05)	81.30

than the SVM. We also tried $n = 2000$ and $n = 3000$ the time of the kernel DWD was 393.60 seconds and 1131.50 seconds, respectively. We also found that the kernel SVM did not run for $n = 2000$ and $n = 3000$. When n is very large (like millions), one can use a simple divide-and-conquer strategy: randomly partition the dataset into K parts and fit DWD on each of the K subsets; the final result is the average of these K independently fitted DWD classifiers. This strategy has been examined and analyzed for the kernel SVM by Hsieh et al. (2014) and kernel ridge regression by Zhang et al. (2015)

2.5.4 Benchmark data examples

We examined the performance of `kerndwd` on 16 UCI benchmark data sets. These real data examples have various combinations of sample size and dimension. We compared the SVM, standard DWD ($q = 1$) and DWD with a data-driven q, under both linear and Gaussian kernels. We randomly split each data into a training and a test set with a ratio

$2:1$. We conducted five-fold cross-validation on the training set to tune each competing method. In particular, we tuned the pair of (q, λ) for DWD with a data-driven q, where q was selected from $\{0.01, 1, 10, 10^5\}$. Table 2.4 summarizes the results. We observe that kernel DWD with a data-driven q is the best on seven data sets and the kernel SVM is the best on four data sets. On some data sets the linear SVM and linear DWD with a data-driven q can outperform the Gaussian kernel counterparts. If we compare only kernel SVM and kernel DWD with a data-driven q, we see that the latter has a lower mis-classification rate in ten examples and is significantly faster in 14 examples. The standard DWD with $q = 1$ is the fastest to compute, but exploring a data-driven q in DWD can lead noticeable improvement in the prediction accuracy with affordable extra computing time.

In Table 2.5, we compared the kernel DWD with another four commonly used classifiers: gradient boosting machines (implemented in the R package `gbm` (Ridgeway, 2017)), random forest (R package `randomForest` (Liaw and Wiener, 2002)), linear discriminant analysis (LDA, R package `MASS` (Venables and Ripley, 2002)), and deep neural net (R package `mxnet` (Chen et al., 2015)). Out of 16 benchmark data, kernel DWD has the least prediction error on 10 examples, both random forest and deep neural net have the least on two cases, and both GBM and LDA have the least on one data set.

Table 2.4: The mean mis-classification rates and computation time (in second) for the SVM, the standard DWD ($q = 1$), and the DWD with a data-driven q on 16 benchmark data. Each data set was split into a training set, to train and to tune, and a test set, to evaluate accuracy. Averaged over 50 runs, the method with the lowest classification error is marked by a black box. The standard error of the mean is given in parentheses. All the computation time includes tuning the parameters. Computations were conducted on a single processor Intel(R) Xeon(R) CPU E5-2660@2.60GHz.

data	n	p	kernel	SVM			$\text{DWD}_{q=1}$			$\text{DWD}_{\text{data-driven } q}$		
				error (%)		time	error (%)		time	error (%)		time
arrhythmia	452	191	linear	**23.67**	(0.48)	35.39	24.27	(0.44)	3.16	24.05	(0.44)	8.00
			Gaussian	23.77	(0.43)	36.81	25.23	(0.52)	1.70	24.49	(0.53)	5.19
australian	690	14	linear	13.75	(0.29)	357.71	13.95	(0.24)	0.10	14.10	(0.29)	0.27
			Gaussian	13.96	(0.29)	12.53	14.30	(0.26)	3.80	**13.45**	(0.27)	15.50
banknote	1372	4	linear	1.01	(0.05)	11.55	2.39	(0.09)	0.20	2.28	(0.09)	0.43
			Gaussian	0.06	(0.02)	20.63	0.39	(0.06)	20.67	**0.00**	(0.00)	52.33
biodeg	1055	41	linear	13.32	(0.23)	501.25	14.67	(0.24)	0.61	14.59	(0.21)	1.42
			Gaussian	**12.36**	(0.22)	36.49	14.57	(0.26)	9.97	13.03	(0.22)	34.67
bupa	345	6	linear	**31.63**	(0.50)	21.40	34.82	(0.75)	0.04	33.22	(0.62)	0.11
			Gaussian	32.23	(0.48)	6.56	32.14	(0.63)	0.92	31.70	(0.54)	6.05
chess	3196	37	linear	3.03	(0.15)	877.39	3.92	(0.08)	2.23	3.92	(0.08)	4.32
			Gaussian	0.93	(0.04)	410.23	5.01	(0.10)	192.73	**0.92**	(0.04)	364.44
heart	270	13	linear	16.53	(0.52)	34.91	16.44	(0.47)	0.05	**15.60**	(0.49)	0.13
			Gaussian	16.69	(0.56)	5.08	16.51	(0.46)	0.63	16.31	(0.48)	2.01
hepatitis	112	18	linear	15.89	(0.73)	5.27	15.03	(0.83)	0.07	15.62	(0.77)	0.14
			Gaussian	15.14	(0.66)	4.07	15.35	(0.85)	0.14	**14.22**	(0.74)	0.37
hungarian	261	10	linear	19.43	(0.47)	20.46	20.78	(0.59)	0.04	**19.26**	(0.47)	0.09
			Gaussian	19.54	(0.54)	4.75	20.99	(0.54)	0.54	19.29	(0.64)	2.71
LSVT	126	309	linear	17.43	(0.94)	15.05	16.33	(0.88)	0.14	17.00	(0.80)	0.30
			Gaussian	**15.52**	(0.76)	16.77	17.33	(0.91)	0.19	16.62	(0.84)	0.51
musk	476	166	linear	17.06	(0.43)	33.59	17.96	(0.53)	2.44	17.42	(0.49)	6.27
			Gaussian	**7.77**	(0.33)	32.81	13.18	(0.46)	1.92	8.03	(0.40)	5.67
parkinsons	195	22	linear	13.57	(0.55)	14.07	14.12	(0.56)	0.09	13.51	(0.57)	0.23
			Gaussian	8.68	(0.55)	4.70	12.86	(0.72)	0.33	**8.46**	(0.69)	1.01
sonar	208	60	linear	25.97	(0.66)	7.55	25.65	(0.75)	0.38	25.59	(0.75)	0.94
			Gaussian	**15.65**	(0.56)	7.91	20.67	(0.76)	0.39	18.29	(0.60)	1.23
spectf	80	44	linear	26.62	(1.02)	4.75	31.31	(1.91)	0.10	29.54	(1.41)	0.29
			Gaussian	22.54	(1.03)	5.01	25.08	(1.32)	0.10	**22.38**	(0.88)	0.25
valley	606	100	linear	4.30	(0.18)	16.45	4.36	(0.21)	1.07	4.13	(0.20)	2.24
			Gaussian	1.40	(0.11)	29.11	3.41	(0.19)	3.26	**0.85**	(0.10)	8.33
vertebral	310	6	linear	**14.83**	(0.42)	8.33	16.76	(0.53)	0.06	16.68	(0.51)	0.14
			Gaussian	16.50	(0.46)	5.33	17.57	(0.49)	0.82	16.60	(0.50)	4.66

Table 2.5: The mean mis-classification rates and computation time (in second) for the DWD with a data-driven q and four commonly used classifiers on 16 benchmark data. Each data set was split into a training set, to train and to tune, and a test set, to evaluate accuracy. Averaged over 50 runs, the method with the lowest classification error is marked by a black box. The standard error of the mean is given in parentheses.

data	n	p	$\text{DWD}_{\text{data-driven } q}$		gradient boosting		random forest		LDA		deep neural net	
arrhythmia	452	191	24.49	(0.53)	44.41	(1.59)	**22.13**	(0.96)	36.71	(1.14)	27.47	(0.74)
australian	690	14	13.45	(0.27)	13.46	(0.58)	**13.02**	(0.50)	14.31	(0.57)	14.43	(0.49)
banknote	1372	4	**0.00**	(0.00)	3.60	(0.23)	1.00	(0.10)	2.41	(0.12)	0.24	(0.09)
biodeg	1055	41	**13.03**	(0.22)	15.55	(0.37)	14.84	(0.36)	14.96	(0.28)	14.05	(0.40)
bupa	345	6	31.70	(0.54)	40.21	(1.31)	31.30	(1.06)	32.84	(0.88)	**29.35**	(0.61)
chess	3196	37	**0.92**	(0.04)	48.29	(0.44)	2.27	(0.32)	5.93	(0.21)	1.44	(0.11)
heart	270	13	16.31	(0.48)	**15.56**	(0.79)	17.78	(0.77)	16.24	(0.65)	19.61	(0.97)
hepatitis	112	18	**14.22**	(0.74)	16.47	(1.26)	15.83	(1.03)	17.89	(0.98)	19.05	(1.15)
hungarian	261	10	19.29	(0.64)	21.62	(0.76)	22.71	(0.82)	**19.27**	(0.51)	19.60	(0.59)
LSVT	126	309	**16.62**	(0.84)	32.88	(1.48)	19.27	(1.47)	26.30	(1.80)	18.81	(1.26)
musk	476	166	**8.03**	(0.40)	18.35	(0.76)	12.66	(0.55)	22.06	(0.71)	11.55	(0.50)
parkinsons	195	22	**8.46**	(0.69)	24.91	(1.15)	11.21	(1.09)	12.53	(0.80)	12.85	(0.88)
sonar	208	60	**18.29**	(0.60)	24.02	(2.45)	20.15	(1.24)	27.47	(1.21)	22.90	(1.04)
spectf	80	44	**22.38**	(0.88)	30.04	(2.62)	23.99	(1.76)	40.48	(2.15)	27.88	(1.87)
valley	606	100	**0.85**	(0.10)	9.15	(2.62)	1.67	(0.35)	5.07	(0.33)	3.02	(0.40)
vertebral	310	6	16.60	(0.50)	15.30	(0.73)	16.32	(0.81)	17.48	(0.88)	**14.51**	(0.65)

2.6 Discussion

In the present chapter we have considered the standard classification problem under the 0-1 loss. In many applications we may face the so-called non-standard classification problems. For example, observed data may be collected via biased sampling and/or we need to consider unequal costs for different types of mis-classification. Qiao et al. (2010) introduced a weighted DWD to handle the non-standard classification problem, which follows the treatment of the non-standard SVM in Lin et al. (2002). Qiao et al. (2010) defined the weighted DWD:

$$\min_{\beta_0,\boldsymbol{\beta}} \left[\sum_{i=1}^{n} w(y_i) \left(\frac{1}{r_i} + c\xi_i \right) \right], \text{ subject to } r_i = y_i(\beta_0 + \boldsymbol{x}_i^T\boldsymbol{\beta}) + \xi_i \geq 0 \text{ and } \boldsymbol{\beta}^T\boldsymbol{\beta} = 1, \tag{2.17}$$

which can be further generalized to the weighted kernel DWD:

$$\min_{\beta_0,\boldsymbol{\alpha}} \boldsymbol{C}_w(\beta_0, \boldsymbol{\alpha}) \equiv \min_{\beta_0,\boldsymbol{\alpha}} \left[\frac{1}{n} \sum_{i=1}^{n} w(y_i) V_q \left(y_i(\beta_0 + \boldsymbol{K}_i^T\boldsymbol{\alpha}) \right) + \lambda \boldsymbol{\alpha}^T \boldsymbol{K} \boldsymbol{\alpha} \right].$$

Qiao et al. (2010) gave the expressions for $w(y_i)$ for various non-standard classification problems. Qiao et al. (2010) solved the weighted DWD with $q = 1$ (2.17) based on the second-order cone programming. The MM procedure developed in this chapter can easily accommodate the weight factors $w(y_i)$'s to solve the weighted DWD and weighted kernel DWD. We have implemented the weighted DWD in the R package `kerndwd`.

Chapter 3

Sparse DWD for High-Dimensional Classification

In this chapter, we propose sparse penalized DWD for high-dimensional classification. The state-of-the-art algorithm for solving the standard DWD is based on second-order cone programming, however such an algorithm does not work well for the sparse penalized DWD with high-dimensional data. In order to overcome the challenging computation difficulty, we develop a very efficient algorithm to compute the solution path of the sparse DWD at a given fine grid of regularization parameters. We implement the algorithm in a publicly available R package `sdwd`. We conduct extensive numerical experiments to demonstrate the computational efficiency and classification performance of our method.

3.1 Introduction

The support vector machine (SVM) (Vapnik, 1995) is a widely used modern classification method. In the standard binary classification problem, training dataset consists of n pairs, $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$, where $\boldsymbol{x}_i \in \mathbb{R}^p$ and $y_i \in \{-1, 1\}$. The linear SVM seeks a hyperplane $\{\boldsymbol{x} : \beta_0 + \boldsymbol{x}^T\boldsymbol{\beta} = 0\}$ which maximizes the smallest margin of all data points:

$$\begin{aligned} &\underset{\beta_0,\boldsymbol{\beta}}{\operatorname{argmax}} \ \min_i d_i, \\ &\text{subject to } \ d_i = y_i(\beta_0 + \boldsymbol{x}_i^T\boldsymbol{\beta}) + \eta_i \geq 0, \ \forall i, \end{aligned}$$

$$\eta_i \geq 0,\ \forall i,\ \sum_{i=1}^{n} \eta_i \leq c,\ ||\boldsymbol{\beta}||_2^2 = 1, \tag{3.1}$$

where d_i is defined as the *margin* of the ith data point, η_i's are slack variables introduced to ensure all margins non-negative, and $c > 0$ is a tuning parameter controlling the overlap. By using a kernel trick, the SVM can also produce nonlinear decision boundaries by fitting an optimal separating hyperplane in an extended kernel feature space.

Marron et al. (2007) noticed that when the SVM is applied on some data with $n < p$, many data points lie on two hyperplanes parallel to the decision boundary. Marron et al. (2007) referred to this phenomenon as *data pilling* and claimed that the data pilling can "affect the generalization performance of SVM". To overcome this issue, Marron et al. (2007) proposed a new method named the distance weighted discrimination (DWD), which finds a separating hyperplane minimizing the sum of the inverse margins of all data points:

$$\begin{aligned} &\underset{\beta_0,\boldsymbol{\beta}}{\operatorname{argmin}} \sum_i 1/d_i, \\ \text{subject to } &\ d_i = y_i(\beta_0 + \boldsymbol{x}_i^T\boldsymbol{\beta}) + \eta_i \geq 0,\ \forall i, \\ &\ \eta_i \geq 0,\ \forall i,\ \sum_i \eta_i \leq c,\ ||\boldsymbol{\beta}||_2^2 = 1. \end{aligned}$$

The initial version of Marron et al. (2007) also mentioned the sum of the inverse margins $\sum_i 1/d_i$ could be also replaced by $\sum_i 1/d_i^q$, the qth power of the inverse margins, and this generalized version was used as the definition of the DWD in Hall et al. (2005). Marron et al. (2007) asserted the DWD can avoid the data piling and thereby improve the generalizability. One example [see the group 2 of Figure 3 in Marron et al. (2007)] shows that the DWD has about 5% prediction error whereas the SVM does 15%. Enhancement of the DWD over the SVM can also be exemplified in Hall et al. (2005) through a novel geometric view. As for the computation of the DWD, Marron et al. (2007) observed that the DWD is an application of the second-order cone programming and thus can be solved by the primal-dual interior-point methods. The algorithm has been implemented in both MATLAB code (Marron, 2013) and an R package `DWD` (Huang et al., 2012). Other notable developments on DWD include weighted DWD (Qiao et al., 2010), multiclass DWD (Huang et al., 2013), and distance weighted SVM (Qiao and Zhang , 2015a) which is a combination of DWD

and SVM.

In this chapter we focus on classification with high-dimensional data where the number of covariates is much larger than the sample size. The standard SVM and DWD are not suitable tools for high-dimensional classification for two reasons. First, based on the scientific hypothesis that only a few important variables affect the outcome, a good classifier for high-dimensional classification should have the ability to select important variables and discard irrelevant ones. However, the standard SVM and DWD use all variables and do not conduct variable selection. Second, because these two classifiers use all variables, they may have very poor classification performance. As explained in Fan and Fan (2008), the bad performance is caused by the error accumulation when estimating too many noise variables in the classifier. Because of these two considerations, sparse classifiers are generally preferred for high-dimensional classification. In the literature, some penalties have been applied to the SVM to produce sparse SVMs such as ℓ_1 SVM (Bradley and Mangasarian, 1998; Zhu et al., 2004), SCAD SVM (Zhang et al., 2006), and elastic-net penalized SVM (Wang et al., 2006).

In this chapter we consider sparse penalized DWD for high dimensional classification. The standard DWD uses the ℓ_2 penalty and can be solved as a second-order cone programming problem. However, sparse DWD is computationally more challenging and requires a different computing algorithm. To cope with the computational challenges associated with the sparse penalty and high-dimensionality, we derive an efficient algorithm to solve sparse DWD by combining majorization-minimization principle and coordinate-descent. We have implemented this algorithm in an R package `sdwd`. To give a quick demonstration here, we use the prostate cancer data (Singh et al., 2002, 102 observations and 6033 genes) as an example. The left panel of Figure 3.1 depicts the solution paths of the elastic-net penalized DWD, and `sdwd` only took 0.453 second to compute the whole solution path. As comparison, we also used the code in Wang et al. (2006) to compute the solution path of elastic-net penalized SVM. We observed that the timing of sparse SVM was about 290 times larger than that of sparse DWD.

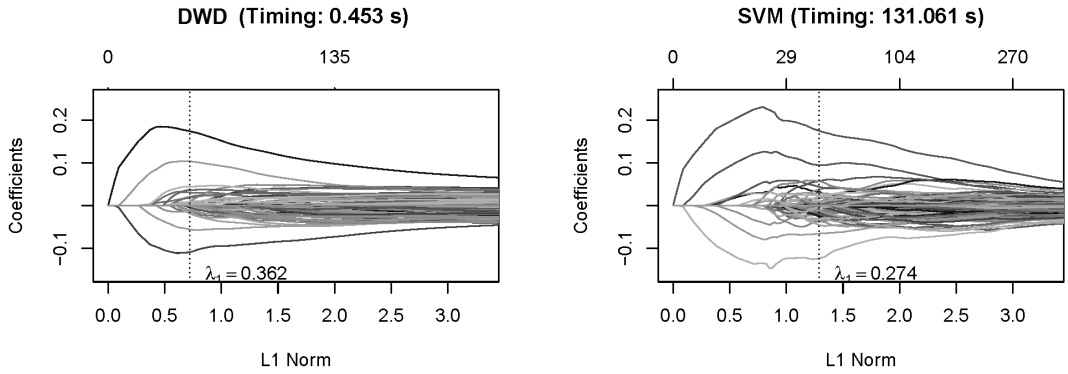

Figure 3.1: The solution paths for the prostate data ($n = 102$, $p = 6033$) using elastic-net DWD and elastic-net SVM. In every method, λ_2 is fixed to be 1. The dashed vertical lines indicate the λ_1 selected by the five-folder cross validation. Both computation time are averaged over 10 runs.

3.2 Sparse DWD

In this section we present several variants of sparse penalized DWD. Our formulation follows ℓ_1 SVM (Zhu et al., 2004). Thus, we first review the derivation process of ℓ_1 SVM. The standard SVM (3.1) is often rephrased as the following quadratic programming problem (Hastie et al., 2009):

$$\begin{aligned} &\underset{\beta_0,\boldsymbol{\beta}}{\operatorname{argmin}} \ ||\boldsymbol{\beta}||_2^2 \\ \text{subject to} \ \ &y_i(\beta_0 + \boldsymbol{x}_i^T\boldsymbol{\beta}) + \eta_i \geq 1, \ \forall i, \\ &\eta_i \geq 0, \ \forall i, \ \sum_{i=1}^{n} \eta_i \leq c. \end{aligned}$$

Moreover, the above constrained minimization problem has an equivalent *loss-plus-penalty* formulation (Hastie et al., 2009):

$$\underset{\beta_0,\boldsymbol{\beta}}{\operatorname{argmin}} \frac{1}{n}\sum_{i=1}^{n}\left[1 - y_i(\beta_0 + \mathbf{x}_i^T\boldsymbol{\beta})\right]_+ + \frac{\lambda_2}{2}||\boldsymbol{\beta}||_2^2.$$

The loss function $[1-t]_+ = \max(1-t,\ 0)$ is the so-called hinge loss in the literature. For the high-dimensional setting, the standard SVM uses all variables because of the ℓ_2

norm penalty used therein. As a result, its performance can be very poor. Zhu et al. (2004) proposed ℓ_1-norm SVM to fix this issue:

$$\underset{\beta_0,\boldsymbol{\beta}}{\operatorname{argmin}} \frac{1}{n}\sum_{i=1}^{n}\left[1-y_i(\beta_0+\boldsymbol{x}_i^T\boldsymbol{\beta})\right]_+ + \lambda_1||\boldsymbol{\beta}||_1.$$

Similarly, we can propose ℓ_1 penalized DWD. It has been shown that the standard DWD also has a *loss-plus-penalty* formulation (Liu et al., 2011):

$$\underset{\beta_0,\boldsymbol{\beta}}{\operatorname{argmin}} \frac{1}{n}\sum_{i=1}^{n} V\left(y_i(\beta_0+\boldsymbol{x}_i^T\boldsymbol{\beta})\right) + \frac{\lambda_2}{2}||\boldsymbol{\beta}||_2^2,$$

where the loss function is given by

$$V(u)=\begin{cases}1-u, & \text{if } u\le 1/2,\\ 1/(4u), & \text{if } u>1/2.\end{cases}$$

Similar to ℓ_1 SVM, we replace the ℓ_2 norm penalty with the ℓ_1 norm penalty in order to achieve sparsity in the DWD classifier. Hence, ℓ_1 DWD is defined by

$$\left(\hat{\beta}_0(\text{lasso}),\hat{\boldsymbol{\beta}}(\text{lasso})\right) = \underset{\beta_0,\boldsymbol{\beta}}{\operatorname{argmin}} \frac{1}{n}\sum_{i=1}^{n} V\left(y_i(\beta_0+\boldsymbol{x}_i^T\boldsymbol{\beta})\right) + \lambda_1||\boldsymbol{\beta}||_1.$$

The lasso penalized DWD classification rule is $\text{Sign}(\hat{\beta}_0(\text{lasso})+\boldsymbol{x}^T\hat{\boldsymbol{\beta}}(\text{lasso}))$. The above *loss-plus-penalty* formulation of sparse DWD is not new. For example, Zhang and Lin (2013) wrote a review paper of classification methods and mentioned the sparse DWD idea in section 4.4 of their paper, although no further technical details were given.

Besides the ℓ_1 norm penalty, we also consider the elastic-net penalty (Zou and Hastie, 2005). It is now well-known that the elastic-net often outperforms the lasso (ℓ_1 norm penalty) in prediction. Wang et al. (2006) studied elastic-net penalized SVM (DrSVM) and showed that the DrSVM performs better than ℓ_1 norm SVM. Likewise, we propose

elastic-net penalized DWD:

$$\left(\hat{\beta}_0(\text{enet}), \hat{\boldsymbol{\beta}}(\text{enet})\right) = \underset{\beta_0,\boldsymbol{\beta}}{\text{argmin}} \frac{1}{n}\sum_{i=1}^{n} V(y_i(\beta_0 + \boldsymbol{x}_i^T\boldsymbol{\beta})) + P_{\lambda_1,\lambda_2}(\boldsymbol{\beta}), \tag{3.2}$$

where

$$P_{\lambda_1,\lambda_2}(\boldsymbol{\beta}) = \sum_{j=1}^{p}\left(\lambda_1|\beta_j| + \frac{\lambda_2}{2}\beta_j^2\right).$$

Elastic-net penalized DWD classification rule is $\text{Sign}(\hat{\beta}_0(\text{enet}) + \boldsymbol{x}^T\hat{\boldsymbol{\beta}}(\text{enet}))$. Both λ_1 and λ_2 are important tuning parameters for regularization. In practice, λ_1 and λ_2 are chosen from finite grids by using validation or cross-validation.

A further refinement of the elastic-net penalty is the adaptive elastic-net penalty (Zou and Zhang, 2009) where we replace the ℓ_1 (lasso) penalty with the adaptive ℓ_1 (lasso) penalty (Zou, 2006). The adaptive lasso penalty produces estimators with the oracle properties. The adaptive elastic-net enjoys the benefits of elastic-net and adaptive lasso. After fitting the elastic-net penalized DWD, we further consider the adaptive elastic-net penalized DWD:

$$\left(\hat{\beta}_0(\text{aenet}), \hat{\boldsymbol{\beta}}(\text{aenet})\right) = \underset{\beta_0,\boldsymbol{\beta}}{\text{argmin}} \frac{1}{n}\sum_{i=1}^{n} V(y_i(\beta_0+\boldsymbol{x}_i^T\boldsymbol{\beta}))+\sum_{j=1}^{p}\left(\lambda_1\hat{\omega}_j|\beta_j| + \frac{\lambda_2}{2}\beta_j^2\right),$$

and the adaptive weights are computed by

$$\hat{\omega}_j = (|\hat{\beta}_j(\text{enet})| + 1/n)^{-1},$$

where $\hat{\beta}_j(\text{enet})$ is the solution of β_j in problem (3.2). Adaptive elastic-net penalized DWD classification rule is $\text{Sign}(\hat{\beta}_0(\text{aenet}) + \boldsymbol{x}^T\hat{\boldsymbol{\beta}}(\text{aenet}))$.

3.3 Computation

In this section, we propose an intuitive but efficient algorithm for computing the solution paths of sparse DWD. Our algorithm uses the generalized coordinate descent (GCD) proposed by Yang and Zou (2013). We introduce the algorithm in Section 3.3.1, the implemen-

tation in Section 3.3.2, and the strict descent property in section Section 3.3.3, respectively. The same algorithm solves all ℓ_1, elastic-net, and adaptive elastic-net penalized DWDs, while only the elastic-net is focused in the discussion for the sake of presentation.

3.3.1 Derivation of the algorithm

Without loss of generality, we assume that the variables $\boldsymbol{x}_j$ are standardized: $\sum_{i=1}^n x_{ij} = 0, \frac{1}{n}\sum_{i=1}^n x_{ij}^2 = 1$, for $j = 1, \ldots, p$. We fix λ_1 and λ_2 and let $u_i = y_i(\tilde{\beta}_0 + \boldsymbol{x}_i^T\tilde{\boldsymbol{\beta}})$. We focus on β_j's first. For each β_j, we define the coordinate-wise update function:

$$F(\beta_j|\tilde{\boldsymbol{\beta}}, \tilde{\beta}_0) = \frac{1}{n}\sum_{i=1}^n V\left(u_i + y_i x_{ij}(\beta_j - \tilde{\beta}_j)\right) + p_{\lambda_1,\lambda_2}(\beta_j). \tag{3.3}$$

Then the standard coordinate descent algorithm suggests cyclically updating

$$\hat{\beta}_j = \underset{\beta_j}{\operatorname{argmin}}\, F(\beta_j|\tilde{\beta}_0, \tilde{\boldsymbol{\beta}}) \tag{3.4}$$

for each $j = 1, \ldots, p$. However, problem (3.4) does not have a closed-form solution. The GCD algorithm solves this issue by adopting the MM principle (Hunter and Lange, 2004). We approximate the F function by a quadratic function

$$Q(\beta_j|\tilde{\boldsymbol{\beta}}, \tilde{\beta}_0) = \frac{\sum_{i=1}^n V(u_i)}{n} + \frac{\sum_{i=1}^n V'(u_i)y_i x_{ij}}{n}(\beta_j - \tilde{\beta}_j) + 2(\beta_j - \tilde{\beta}_j)^2 + p_{\lambda_1,\lambda_2}(\beta_j). \tag{3.5}$$

Then we update $\tilde{\beta}_j$ by $\tilde{\beta}_j^{\text{new}}$, the closed-form minimizer of problem (3.5):

$$\tilde{\beta}_j^{\text{new}} = \frac{S\left(M\tilde{\beta}_j - \frac{1}{n}\sum_{i=1}^n V'(u_i)y_i x_{ij}, \lambda_1\right)}{4 + \lambda_2}, \tag{3.6}$$

where $S(z, r) = \text{sign}(z)(|z| - r)_+$ is the soft-thresholding operator (Donoho and Johnston, 1994) and $\omega_+ = \max(\omega, 0)$ is the positive part of ω.

With the intercept similarly updated, Algorithm 1 summarizes the details of the GCD algorithm.

Algorithm 1 *The GCD algorithm for sparse DWD*

1. Initialize $(\tilde{\beta}_0, \tilde{\boldsymbol{\beta}})$.
2. Cyclic coordinate descent, for $j = 1, 2, \ldots, p$:
   (a) Compute $u_i = y_i(\tilde{\beta}_0 + \boldsymbol{x}_i^T \tilde{\boldsymbol{\beta}})$.
   (b) Compute $\tilde{\beta}_j^{\text{new}} = \frac{1}{4+\lambda_2} \cdot S\left(4\tilde{\beta}_j - \frac{1}{n}\sum_{i=1}^n V'(u_i) y_i x_{ij}, \lambda_1\right)$.
   (c) Set $\tilde{\beta}_j = \tilde{\beta}_j^{\text{new}}$.
3. Update the intercept term:
   (a) Compute $u_i = y_i(\tilde{\beta}_0 + \boldsymbol{x}_i^T \tilde{\boldsymbol{\beta}})$.
   (b) Compute $\tilde{\beta}_0^{\text{new}} = \tilde{\beta}_0 - \sum_{i=1}^n V'(u_i) y_i/(4n)$.
   (c) Set $\tilde{\beta}_0 = \tilde{\beta}_0^{\text{new}}$.
4. Repeat steps 2-3 until convergence of $(\tilde{\beta}_0, \tilde{\boldsymbol{\beta}})$.

3.3.2 Implementation

We have implemented Algorithm 1 in an R package `sdwd`. We exploit the warm-start, the strong rule, and the active set trick to increase the algorithm speeding. In our implementation, λ_2 is pre-chosen and we compute the solution path as λ_1 varies.

First, we adopt the warm-start to lead to a faster and more stable algorithm (Friedman et al., 2007). We compute the solutions at a grid of K decreasing λ_1 values, starting at the smallest λ_1 value such that $\tilde{\boldsymbol{\beta}} = 0$. Denote these grid points by $\lambda_1^{[1]}, \ldots, \lambda_1^{[K]}$. With the warm-start trick, we can use the solution at $\lambda_1^{[k]}$ as the initial value (the warm-start) to compute the solution at $\lambda_1^{[k+1]}$.

Specifically, to find $\lambda_1^{[1]}$, we fit a model with a sufficiently large λ_1 and thus $\tilde{\boldsymbol{\beta}} = 0$. Let $\hat{\beta}_0$ be the estimate of the intercept. By the Karus–Kuhn–Tucker (KKT) conditions, $\frac{1}{n}\max_j \left|\sum_{i=1}^n V'(\hat{\beta}_0) y_i x_{ij})\right| \leq \lambda_1$, so we can choose

$$\lambda_1^{[1]} = \frac{1}{n}\max_j \left|\sum_{i=1}^n V'(\hat{\beta}_0) y_i x_{ij})\right|.$$

Generally, we use $K = 100$, and $\lambda_1^{[100]} = \epsilon\lambda_1^{[1]}$, where $\epsilon = 10^{-4}$ when $n < p$ and $\epsilon = 10^{-2}$ otherwise. All the other grid points are placed to uniformly distribute on a log scale.

Second, we follow the strong rule (Tibshirani et al., 2010) to improve the computational speed. Suppose $\tilde{\boldsymbol{\beta}}^{[k]}$ and $\tilde{\beta}_0^{[k]}$ are the solutions at $\lambda_1^{[k]}$. After we solve $\tilde{\boldsymbol{\beta}}^{[k]}$ and $\tilde{\beta}_0^{[k]}$, the strong rule claims that any $j \in \{1, \ldots, p\}$ satisfying

$$\left| \frac{1}{n} \sum_{i=1}^{n} V'(y_i(\hat{\beta}_0^{[k]} + \boldsymbol{x}_i^T \hat{\boldsymbol{\beta}}^{[k]})) y_i x_{ij} \right| < 2\lambda_1^{[k+1]} - \lambda_1^{[k]} \tag{3.7}$$

is likely to be inactive at $\lambda_1^{[k+1]}$, i.e., $\hat{\beta}_j^{[k+1]} = 0$. Let $\mathcal{D}$ be the collection of j which satisfies inequality (3.7), and its compliment $\mathcal{D}^C = \{1, \ldots, p\} \backslash \mathcal{D}$. We call $\mathcal{D}^C$ the survival set. If the strong rule guesses correctly, the variables contained in $\mathcal{D}$ are discarded, and we only apply Algorithm 1 to repeat the coordinate descent in the survival set $\mathcal{D}^C$. After computing the solution $\hat{\beta}_0$ and $\hat{\boldsymbol{\beta}}$, we need to check whether some variables are incorrectly discarded. We check this by the KKT condition,

$$\left| \frac{1}{n} \sum_{i=1}^{n} V'(y_i(\hat{\beta}_0 + \boldsymbol{x}_i^T \hat{\boldsymbol{\beta}})) y_i x_{ij} \right| \leq \lambda_1. \tag{3.8}$$

If no $j \in \mathcal{D}$ violates condition (3.8), $\hat{\beta}_0$ and $\hat{\boldsymbol{\beta}}$ are the solutions at $\lambda_1^{[k+1]}$. We rephrase them as $\tilde{\beta}_0^{[k+1]}$ and $\tilde{\boldsymbol{\beta}}^{[k+1]}$. Otherwise, any incorrectly discarded variable should be added to the survival set $\mathcal{D}^C$. We update $\mathcal{D}$ by $\mathcal{D} = \mathcal{D}/U$ where

$$U = \left\{ j : j \in \mathcal{D} \text{ and } \left| \frac{1}{n} \sum_{i=1}^{n} V'(y_i(\hat{\beta}_0 + x_i^T \hat{\boldsymbol{\beta}})) y_i x_{ij} \right| > \lambda_1 \right\}.$$

After each update of $\mathcal{D}$, some incorrectly discarded variables are added back to the survival set.

Third, the active set is also used to boost the algorithm speed. After we apply Algorithm 1 on the survival set $\mathcal{D}^C$, we only apply the coordinate descent on a subset S of $\mathcal{D}^C$ till convergence, where $S = \left\{ j : j \in \mathcal{D}^C \text{ and } \beta_j \neq 0 \right\}$. Then another cycle of coordinate descent is run on $\mathcal{D}^C$ to investigate if the active set S changes. We finish the algorithm if no changes in S; otherwise, we update the active set S and repeat the process.

In Algorithm 1, the margin u_i can be updated conveniently: if β_j is updated by β_j^{new}, we update u_i by $u_i + y_i x_{ij}(\beta_j^{\text{new}} - \beta_j)$.

Last, the default convergence rule in `sdwd` is $4(\tilde{\beta}_j^{\text{new}} - \tilde{\beta}_j)^2 < 10^{-8}$ for all $j = 0, 1, \ldots, p$.

3.3.3 The strict descent property of Algorithm 1

Yang and Zou (2013) showed the GCD algorithm enjoys descent property. In this section, we also show the GCD algorithm has a stronger statement, the strict descent property, when the GCD is used to solve sparse DWD. We first elaborate the following majorization result, whose proof is given in Appendix C.

Lemma 3.1

$F(\beta_j|\tilde{\boldsymbol{\beta}}, \tilde{\beta}_0)$ is the coordinate-wise update function defined in (3.3), and $Q(\beta_j|\tilde{\boldsymbol{\beta}}, \tilde{\beta}_0)$ is the surrogate function defined in (3.5). We have (3.9) and (3.10):

$$F(\beta_j|\tilde{\boldsymbol{\beta}}, \tilde{\beta}_0) = Q(\beta_j|\tilde{\boldsymbol{\beta}}, \tilde{\beta}_0), \text{ if } \beta_j = \tilde{\beta}_j, \tag{3.9}$$

$$F(\beta_j|\tilde{\boldsymbol{\beta}}, \tilde{\beta}_0) < Q(\beta_j|\tilde{\boldsymbol{\beta}}, \tilde{\beta}_0), \text{ if } \beta_j \neq \tilde{\beta}_j. \tag{3.10}$$

Given $\tilde{\beta}_j^{\text{new}} = \text{argmin}_{\beta_j} Q(\beta_j|\tilde{\beta}_0, \tilde{\boldsymbol{\beta}})$, and assuming $\tilde{\beta}_j^{\text{new}} \neq \tilde{\beta}_j$, (3.9) and (3.10) imply the strict descent property of the GCD algorithm: $F(\tilde{\beta}_j^{\text{new}}|\tilde{\boldsymbol{\beta}}, \tilde{\beta}_0) < F(\tilde{\beta}_j|\tilde{\boldsymbol{\beta}}, \tilde{\beta}_0)$. It is because $F(\tilde{\beta}_j^{\text{new}}|\tilde{\boldsymbol{\beta}}, \tilde{\beta}_0) < Q(\tilde{\beta}_j^{\text{new}}|\tilde{\boldsymbol{\beta}}, \tilde{\beta}_0) < Q(\tilde{\beta}_j|\tilde{\boldsymbol{\beta}}, \tilde{\beta}_0) = F(\tilde{\beta}_j|\tilde{\boldsymbol{\beta}}, \tilde{\beta}_0)$. Note that the original GCD paper only showed $F(\tilde{\beta}_j^{\text{new}}|\tilde{\boldsymbol{\beta}}, \tilde{\beta}_0) \leq F(\tilde{\beta}_j|\tilde{\boldsymbol{\beta}}, \tilde{\beta}_0)$.

The arguments above prove that the objective function F strictly decreases after updating all variables in a cycle, unless the solution does not change after each update. If this is the case, the algorithm stops. We show that the algorithm must stop at the right answer. Assuming $\tilde{\beta}_j = \tilde{\beta}_j^{\text{new}}$ for all j, (3.6) implies:

$$\tilde{\beta}_j = \frac{S(4\tilde{\beta}_j - \frac{1}{n}\sum_{i=1}^n V'(u_i) y_i x_{ij}, \lambda_1)}{4 + \lambda_2}.$$

A straightforward algebra can show that for all j,

$$\begin{aligned} &\frac{1}{n}\sum_{i=1}^{n} V'(u_i)y_i x_{ij} + \lambda_1 \text{sign}(\beta_j) + \lambda_2\beta_j = 0, \quad \text{if } \beta_j \neq 0; \\ &\left|\frac{1}{n}\sum_{i=1}^{n} V'(u_i)y_i x_{ij}\right| \leq \lambda_1, \qquad\qquad\qquad\qquad \text{if } \beta_j = 0, \end{aligned}$$

which is exactly the KKT conditions of the original objective function (3.2). In conclusion, if the objective function does not change after a cycle, the algorithm necessarily converges to the correct solution satisfying the KKT condition.

3.4 Simulation

The simulation in this section aims to support the following three points: (1) sparse DWD has highly competitive prediction accuracy with the sparse SVM and sparse logistic regression; (2) adaptive elastic-net penalized DWD performs the best in variable selection; (3) for the prediction accuracy, no single method among ℓ_1, elastic-net, and adaptive elastic-net penalized DWDs dominate the others in all situations.

In this section, the response variables of all the data are binary. The dimension p of the variables $\boldsymbol{x}_i$ is always 3000. Within each example, our simulated data consist of a training set, an independent validation set, and an independent test set. The training set contains 50 observations: 25 of them are from the positive class and the other 25 from the negative class. Models are fitted on the training data only, and we use an independent validation set of 50 observations to select the tuning parameters: λ_2 is selected from 10^{-4}, 10^{-3}, 10^{-2}, 0.1, 1, 5, and 10; λ_1 is searched along the solution paths. We compared the prediction accuracy (in percentage) on another independent test data set of 20,000 observations.

We followed Marron et al. (2007) to generate the first two examples. In example 1, the positive class is a random sample from $N_p(\boldsymbol{\mu}_+, \boldsymbol{I}_p)$, where $\boldsymbol{I}_p$ is the p by p identity matrix and $\boldsymbol{\mu}_+$ has all zeros except for 2.2 at the first dimension; the negative class is from $N_p(\boldsymbol{\mu}_-, \boldsymbol{I}_p)$ with $\boldsymbol{\mu}_- = -\boldsymbol{\mu}_+$. In example 2, 80% of the data are generated from the same distributions as example 1; for the other 20% of the data, the positive class is drawn from $N_p(\boldsymbol{\mu}_+, \boldsymbol{I}_p)$ and negative class $N_p(-\boldsymbol{\mu}_+, \boldsymbol{I}_p)$ where $\boldsymbol{\mu}_+ = (100, 500, 0, \ldots, 0)$. We

obtained the other three examples following Wang et al. (2006). In example 3, the positive class has a normal distribution with mean $\boldsymbol{\mu}_+$ and covariance $\boldsymbol{\Sigma} = \boldsymbol{I}_{p\times p}$, where $\boldsymbol{\mu}_+$ has 0.7 in the first five covariates and 0 in others; the negative class has the same distribution except for a different mean $\boldsymbol{\mu}_- = -\boldsymbol{\mu}_+$. In example 4 and 5, we consider the cases where the relevant variables are correlated. Two classes have the same distributions except for the covariance,

$$\boldsymbol{\Sigma} = \begin{pmatrix} \boldsymbol{\Sigma}^{\star}_{5\times 5} & \mathbf{0}_{5\times(p-5)} \\ \mathbf{0}_{(p-5)\times 5} & \boldsymbol{I}_{(p-5)\times(p-5)} \end{pmatrix}.$$

In example 4, the diagonal elements of $\boldsymbol{\Sigma}^{\star}$ are 1 and the off-diagonal elements are all equal to 0.7. In example 5, the (i,j)th element of $\boldsymbol{\Sigma}^{\star}$ equals $0.7^{|i-j|}$.

We compared sparse DWD with the sparse SVM and sparse logistic regression. Both DWD and logistic regression use the ℓ_1, the elastic-net and the adaptive elastic-net penalties. We used R packages `sdwd` and `gcdnet` (Yang and Zou, 2013) to compute sparse DWDs and sparse logistic regressions respectively. The ℓ_1 and elastic-net SVMs were solved by using the code from Wang et al. (2006) which does not handle the adaptive elastic-net penalty. Table 3.1 presents the prediction accuracy results. In the first two examples, the ℓ_1 DWD and the ℓ_1 logistic regression perform the best. We attribute this good performance to the only one nonzero variable in the data, despite 20% of outliers in example 2. In example 3, 4, and 5, we increase the number of nonzero variables to five. For all models, the elastic-net and the adaptive elastic-net penalties have similar performance, and both of them dominate the ℓ_1 penalties. Elastic-net DWD produces the least prediction error in example 4 and 5. Table 3.2 compares the variable selection. In all cases, the adaptive elastic-net penalties address all relevant variables with relatively few mistakes. The ℓ_1 penalties share similar performance in the first two examples.

Table 3.1: Comparisons of mis-classification percentage on 300 training data, 300 validation data, and 20,000 test data, based on 200 replicates. The numbers in parentheses are the standard errors. For each example, the methods with the best performance are marked by black boxes.

	DWD			SVM		logistic		
	ℓ_1	enet	aenet	ℓ_1	enet	ℓ_1	enet	aenet
Example 1	**1.42**	1.47	1.44	1.46	1.50	**1.42**	1.46	1.44
Bayes: 1.39	(0.01)	(0.02)	(0.01)	(0.01)	(0.02)	(0.01)	(0.02)	(0.02)
Example 2	1.14	1.15	1.13	1.16	1.16	**1.11**	1.14	1.15
Bayes: 1.11	(0.01)	(0.01)	(0.01)	(0.01)	(0.01)	(0.01)	(0.01)	(0.02)
Example 3	6.41	6.25	6.21	6.45	**6.15**	6.40	6.21	6.22
Bayes: 5.88	(0.03)	(0.03)	(0.03)	(0.04)	(0.03)	(0.03)	(0.03)	(0.03)
Example 4	22.05	**21.48**	21.54	22.03	21.56	22.00	21.54	21.64
Bayes: 21.10	(0.07)	(0.07)	(0.05)	(0.06)	(0.05)	(0.06)	(0.06)	(0.06)
Example 5	18.91	**18.74**	18.75	18.84	18.78	18.81	18.80	18.77
Bayes: 18.03	(0.07)	(0.05)	(0.05)	(0.06)	(0.05)	(0.06)	(0.05)	(0.05)

Table 3.2: Comparisons of the variable selection. C is the number of selected nonzero variables, and IC is the number of zero variables incorrectly selected into the model. The results are the medians over 200 replicates.

	DWD						SVM				logistic					
	ℓ_1		enet		anet		ℓ_1		enet		ℓ_1		enet		aenet	
	C	IC	C	IC	C	IC	C	IC	C	IC	C	IC	C	IC	C	IC
Example 1	1	0	1	2	1	0	1	0	1	4	1	0	1	4.5	1	0
Example 2	1	0	1	0	1	0	1	0	1	1	1	0	1	0	1	0
Example 3	5	0	5	5	5	0	5	0	5	2.5	5	1	5	7	5	0
Example 4	4	1	5	8.5	5	1.5	4	0	5	7	4	1	5	14	5	2
Example 5	4	1	5	3.5	5	0	4	0	5	2	4	1	5	6.5	5	0

3.5 Real Data Examples

In this section we analyze four benchmark data sets. The data Arcene was obtained from Dua, D. and Karra Taniskidou, E. (2017), the breast cancer data from Graham et al. (2010), the LSVT data from Tsanas et al. (2014), and the prostate cancer was from Singh et al. (2002). We randomly split each data with a ratio 1:1 into a training set and a test set. On the training set, we fit sparse DWD with imposing the elastic-net and the adaptive elastic-net penalties. With the same tuning parameter candidates in the simulation, we used a five folder cross validation to find the best pair of (λ_1, λ_2) incurring the least mis-classification rate. Then we investigated the prediction accuracy of the selected model on the test set. As comparisons, we considered the sparse SVM and the sparse logistic regression. Every method was trained and tuned in the same way as the sparse DWD. All numerical experiments were carried out on an Intel Core i7-3770 (3.40 GHz) processor.

In Table 3.3, we reported the average mis-classification percentage on the test set from 200 independent splits. We observe that the classifiers achieving the least error in these four datasets are adaptive elastic-net logistic regression, elastic-net SVM, elastic-net and adaptive elastic-net DWDs. We also find all the differences are not quite large. For sparse DWD, we get the same message as Marron et al. (2007) concluded for standard DWD: "it very often is competitive with the best of the others and sometimes is better." We also notice that the computation of sparse DWD is the fastest in almost all cases. The timing of the SVM is much longer than other methods. A possible explanation is that the SVM uses the non-differentiable hinge loss function which makes the GCD algorithm not suitable for solving the sparse SVM. So far, the best algorithm for the sparse SVM is a LARS type algorithm Wang et al. (2006), which is very different from the GCD algorithm for sparse DWD and logistic regression. It has been observed that coordinate descent may be faster than the LARS algorithm for solving the lasso penalized least squares (Friedman et al., 2007).

Table 3.3: The mean mis-classification percentage and timings (in seconds) for four benchmark datasets. All the timings include the five-folder cross validation. The timings of adaptive elastic-net methods include computing the weights. The numbers in parentheses are the standard errors. For each data, the methods with the best prediction accuracy are marked by black boxes.

	Arcene		Breast		LSVT		Prostate	
	$n = 100, p = 10000$		$n = 42, p = 22283$		$n = 126, p = 309$		$n = 102, p = 6033$	
	error	time	error	time	error	time	error	time
enet DWD	34.43	123.41	26.50	58.40	16.01	8.28	**10.22**	28.18
	(0.56)	(5.16)	(1.00)	(1.90)	(0.34)	(0.23)	(0.30)	(0.95)
aenet DWD	34.60	200.19	26.86	116.12	**15.92**	13.72	10.26	39.25
	(0.57)	(9.24)	(1.00)	(3.78)	(0.34)	(0.29)	(0.26)	(1.24)
enet logistic	34.16	211.18	24.67	145.35	16.96	10.73	10.65	102.19
	(0.58)	(3.40)	(1.00)	(0.74)	(0.37)	(0.18)	(0.29)	(1.56)
aenet logistic	**34.15**	393.03	25.12	290.31	16.93	17.02	10.75	189.44
	(0.57)	(6.52)	(0.87)	(1.47)	(0.37)	(0.29)	(0.29)	(2.84)
enet SVM	35.10	7410.09	**23.95**	567.43	16.27	63.10	10.56	2508.94
	(0.67)	(1465.68)	(1.00)	(15.19)	(0.37)	(0.77)	(0.36)	(0.77)

3.6 Discussion

In this chapter, we have proposed sparse DWD for high-dimensional classification and developed an efficient algorithm to compute its solution path. We have shown that sparse DWD has competitive prediction performance with the sparse SVM and sparse logistic regression and is often faster to compute with the help of our algorithm. Thus, sparse DWD is a valuable addition to the toolbox for high-dimensional classification.

Generalized DWD studied in Chapter 2 minimizes the qth power of the inverse margins. When $q = 1$, it reduces to the usual DWD. For computation considerations, Marron et al. (2007) choose to fix $q = 1$, because it leads to a second-order cone programming problem. We have found that our algorithm can be readily used to solve sparse generalized DWD with any positive q. In our numerical study we tried the generalized DWD with $q = 0.5, 1, 2, 5, 100$ and also tried to use cross-validation to select a data-driven q value. Our numeric results indicated that using different q values does not lead to significant differences in performance.

Chapter 4

Multicategory Kernel DWD for Multiclass Classification

In this chapter, we propose a new multicategory kernel DWD that is defined as a margin-vector optimization problem in a reproducing kernel Hilbert space. This formulation is shown to enjoy Fisher consistency. We develop an accelerated projected gradient descent algorithm to fit multicategory kernel DWD. Simulations and benchmark data applications are used to demonstrate the highly competitive performance of our method, as compared with some popular state-of-the-art multiclass classifiers.

4.1 Introduction

Classification is a task of identifying observations to one of several pre-defined categories, and its applications are extremely diverse, ranging from daily life to frontiers of science and engineering. Two classic examples are detecting spam e-mail based on the message content and categorizing tissues as tumor or benign based on DNA microarray data. Many real-world problems have multicategory responses. Speech recognition has been formulated as a multicategory classification problem to analyze voice input, which enables the translation of spoken language into text and has many promising applications in as court reporting, mobile e-mail, and robotics (Rabiner, 1989; Rabiner and Juang, 1993; Hansen and Hasan, 2015; Yu and Deng, 2016). Speech recognition has also greatly helped people with hearing disturbances (Chen et al., 2016; Takashima et al., 2017; Wang, 2017). Image

classification (Haralick and Shanmugam, 1973; Krizhevsky et al., 2012; Russakovsky et al., 2015) is another hot application, referring to detection of an object in digital images: for instance, satellite remote images have been used to successfully predict earthquakes (Dong and Shan, 2013; Lillesand et al., 2014; Maulik and Chakraborty, 2017), vision-based road detection inspires the study of self-driving vehicles (Xu et al., 2016; Bojarski et al., 2017), and facial expression extraction facilitates interactions between humans and machines (Liu et al., 2012; Barsoum et al., 2016). Besides engineering applications, binary and multicategory classifications are also abundant in biology, climatology, geology, economics, and finance, among many others.

For binary classification, the support vector machine (SVM, Vapnik, 1995) is a commonly used large-margin classifier. Another large margin classifier is the distance weighted discrimination (DWD) proposed by Marron et al. (2007). Although the SVM and DWD are originally designed for binary classification, they can be generalized to multicategory classification problems. Two simple approaches are one-versus-one and one-versus-rest that decompose multicategory classification into a set of multiple binary classification problems (Hastie and Tibshirani, 1998; Hsu and Lin, 2002). In particular, one-versus-one approach solves each of the pairwise two-class problem and predicts the class that wins the most comparisons, but it may suffer from the tie-in-vote issue. One-versus-rest approach alternatively treats each class as positive and all the other classes as negative; however, this approach has shown to be inconsistent in many situations (Lee et al., 2004; Liu, 2007). In addition, error-correcting coding is an information-theoretic approach that turns the multicategory response into a coding matrix; details are seen in Dietterich and Bakiri (1995); James and Hastie (1998); Allwein et al. (2000). Instead of reducing multicategory classification to binary problems, another approach is to propose a unified framework that considers all classes at once. With such a simultaneous fashion, there are several multicategory SVMs developed in Vapnik (1998); Weston and Watkins (1999); Lee et al. (2004), as well as multicategory extension of other large-margin classifiers including import vector machine (Zou and Hastie, 2005), ψ-learning (Liu and Shen, 2006), large-margin unified machines (Zhang and Lin, 2013), and angle-based large-margin classification (Zhang and Liu, 2014; Zhang et al., 2016).

In the context of DWD, Huang et al. (2013) proposed a multiclass generalization of

linear DWD. From methodological and theoretical viewpoint, the linear classifier will be inadequate because the optimal Bayes rule can often be non-linear. However, it is unclear how to generalize the linear multiclass DWD (Huang et al., 2013) to its kernel counterpart. The same difficulty appeared in the development of the original binary linear DWD (Marron et al., 2007), and a kernel binary DWD, that is computationally efficient and theoretically justified, was only recently proposed in Chapter 2. Moreover, the kernel binary DWD in Chapter 2 has been shown to enjoy very competitive classification performance against popular classifiers such as the SVM, random forest, gradient boosting, and k-nearest neighbors, etc. Given its excellent performance for binary classification, it will be interesting and natural to ask whether the DWD idea could also be competitive for multiclass classification. Hence, it is necessary to derive the kernel version of the multicategory DWD in order to handle multiclass classification problems with complex non-linear decision boundaries.

In this chapter, we develop a multicategory kernel DWD by formulating the multicategory DWD in a reproducing kernel Hilbert space (RKHS). Our formulation of multicategory DWD is completely different from the approach (Huang et al., 2013) that generalizes the linear DWD. We used the concept *margin vector* introduced by Zou et al. (2008), where the margin vector is defined to be a multicategory generalization of the margin in binary classification and can be regarded as a proxy of the conditional class probability. With the device of margin vector, we propose multicategory kernel DWD, and we then demonstrate that our proposal is multicategory Fisher-consistent, in the sense that the class with the largest conditional class probability always has the largest margin. To compute multicategory kernel DWD, we present a multicategory representer theorem, and we develop a projected gradient descent algorithm. We further implement the Nesterov's acceleration to improve the rate of convergence, thereby reducing the number of iterations effectively.

To give a quick illustration, Figure 4.1 delineates the decision boundaries of multicategory kernel DWD and the Bayes rule for a simulation example based on mixture Gaussian distributions. Figure 4.1 shows that the Bayes rule has a non-linear decision boundary and our method resembles the Bayes rule. This example clearly reveals the inadequacy of the multicategory linear DWD as well as the excellent performance of our new method.

The rest of the chapter is organized as follows. In Section 2, we briefly review DWD in binary classification and multicategory linear DWD proposed in Huang et al. (2013). Sec-

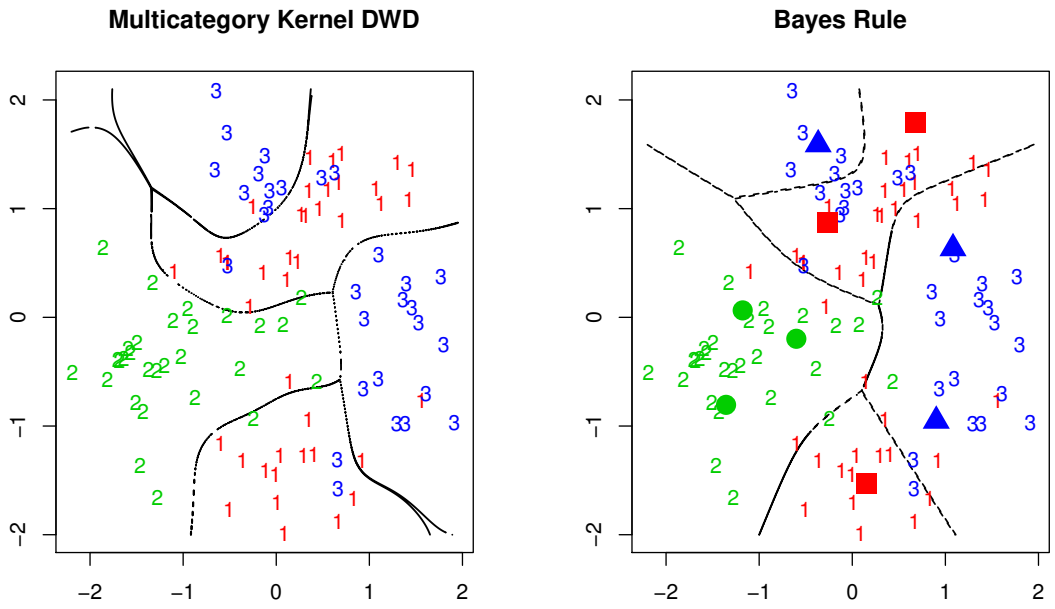

Figure 4.1: Decision boundaries of multicategory kernel DWD (left panel) and the Bayes rule (right panel). We simulated three classes, each of which follows a mixture Gaussian distribution $\frac{1}{3}\sum_{i=1}^{3} N(\boldsymbol{\mu}_i, \tau^2 \boldsymbol{I})$, where $\boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \boldsymbol{\mu}_3$ are three centers independently drawn from standard normal distribution. In the right panel, the centers of class 1, 2, and 3 are depicted as red squares, green circles, and blue triangles, respectively. We set $\tau = 0.4$ and Bayes error is 13.4% in this example. The proposed multicategory kernel DWD is fit based on 100 training data and its misclassification error rate is 15.9%. In contrast, the multicategory linear DWD (Huang et al., 2013) has a misclassification error rate of 34.5%.

tion 3 describes our proposal of multicategory kernel DWD and we explore its multicategory Fisher consistency. In Section 4 we derive an efficient convex optimization algorithm to solve the proposed classifier. Simulations and benchmark data examples are presented in Section 5.

4.2 Review of Distance Weighted Discrimination

Before introducing multicategory kernel DWD, it is necessary to review the basic idea of the original binary DWD. Suppose that a training data set consists of n pairs of observa-

tions, $\{\boldsymbol{x}_i, y_i\}_{i=1}^n$, where $\boldsymbol{x}_i \in \mathbb{R}^p$ and $y_i \in \{-1, 1\}$. Linear DWD seeks a hyperplane $\{\boldsymbol{x} : \hat{\beta}_0 + \boldsymbol{x}^\top \hat{\boldsymbol{\beta}} = 0\}$ where

$$\begin{aligned} &\left(\hat{\beta}_0, \hat{\boldsymbol{\beta}}\right) = \underset{\beta_0, \boldsymbol{\beta}}{\operatorname{argmin}} \sum_{i=1}^n \left(\frac{1}{r_i} + C\xi_i\right), \\ &\text{subject to } r_i = y_i(\beta_0 + \boldsymbol{x}_i^\top \boldsymbol{\beta}) + \xi_i \geq 0, \xi_i \geq 0, \forall i, \|\boldsymbol{\beta}\|_2^2 = 1, \end{aligned} \tag{4.1}$$

where C is a tuning parameter controlling the slack variables ξ_i. DWD predicts the class label of a new observation $\boldsymbol{x}_{\text{new}}$ by $\operatorname{sgn}(\hat{\beta}_0 + \boldsymbol{x}_{\text{new}}^\top \hat{\boldsymbol{\beta}})$. The problem (4.1) was originally solved by second-order cone programming (Marron et al., 2007). Other developments of linear DWD include weighted DWD (Qiao et al., 2010), distance weighted SVM (Qiao and Zhang , 2015a), flexible assortment machines (Qiao and Zhang , 2015b), and sparse DWD proposed in Chapter 3 of this thesis.

Huang et al. (2013) proposed a multicategory linear DWD. Suppose the response of a training data set $\{\boldsymbol{x}_i, y_i\}_{i=1}^n$ has k categories, i.e., $y_i \in \{1, \ldots, k\}$. A vector of discriminant functions $\boldsymbol{f} = (f_1, \ldots, f_k)$ is introduced where each element corresponds to one class. For any new observation $\boldsymbol{x}_{\text{new}}$, the label is predicted by $\hat{y}_{\text{new}} = \operatorname{argmax}_j \hat{f}_j(\boldsymbol{x}_{\text{new}})$, where each $\hat{f}_j(\boldsymbol{x}) = \hat{\beta}_0 + \boldsymbol{x}^\top \hat{\boldsymbol{\beta}}$ and $(\hat{\beta}_{0j}, \hat{\boldsymbol{\beta}}_j)$ are estimated by

$$\begin{aligned} &\min_{\beta_{0j}, \boldsymbol{\beta}_j} \sum_{i=1}^n \sum_{j \neq l} \left(\frac{1}{r_i^{(jl)}} + C\xi_i^{(jl)}\right), \\ &\text{subject to } r_i^{(jl)} = f_j(\boldsymbol{x}_i) - f_l(\boldsymbol{x}_i) + \xi_i^{(jl)}, \text{ for } y_i = j, l \neq j, \\ &\qquad f_j(\boldsymbol{x}_i) = \beta_{0j} + \boldsymbol{x}_i^\top \boldsymbol{\beta}_j, \\ &\qquad r_i^{(jl)} \geq 0, \xi_i^{(jl)} \geq 0, \sum_{j=1}^k \beta_{0j} = 0, \sum_{j=1}^k \boldsymbol{\beta}_j = \mathbf{0}, \|\boldsymbol{\beta}_j\|_2^2 \leq 1. \end{aligned} \tag{4.2}$$

Like the binary linear DWD, the problem (4.2) is solved by second-order cone programming. However, it is unclear how to extend the formulation (4.2) to a reproducing kernel Hilbert space so that one can fit a nonlinear kernel classifier. It is difficult even when the problem (4.2) degenerates to the binary DWD when $k = 2$. Only recently, Chapter 2 of this thesis derived a kernel DWD based on a different formulation of linear DWD.

4.3 A New Multicategory kernel DWD

In this section, we develop a multicategory DWD in an RKHS, and we elucidate its Fisher-consistent property.

4.3.1 Statistical view of distance weighted discrimination

Chapter 2 of this thesis showed that the linear DWD classifier can be equivalently derived from a regularized empirical risk minimization approach as

$$\left(\hat{\beta}_0, \hat{\boldsymbol{\beta}}\right) = \underset{\beta_0, \boldsymbol{\beta}}{\operatorname{argmin}} \left[\frac{1}{n}\sum_{i=1}^{n} \phi\left\{y_i(\beta_0 + \boldsymbol{x}_i^\top \boldsymbol{\beta})\right\} + \lambda \boldsymbol{\beta}^\top \boldsymbol{\beta}\right],$$

where

$$\phi(u) = \begin{cases} 1-u, & \text{if } u \le 1/2, \\ 1/(4u), & \text{if } u > 1/2, \end{cases} \tag{4.3}$$

and the DWD classifer is $\operatorname{sgn}(\hat{\beta}_0 + \boldsymbol{x}^\top \hat{\boldsymbol{\beta}})$. The loss function $\phi(u)$ has also appeared in Qiao et al. (2010); Liu et al. (2011). For the kernel DWD, we formulated kernel DWD in Chapter 2 as $\operatorname{sgn}(\hat{f}(\boldsymbol{x}))$ where $\hat{f}$ is given by

$$\hat{f} = \underset{f \in \mathcal{H}_K}{\operatorname{argmin}} \left[\frac{1}{n}\sum_{i=1}^{n} \phi\left\{y_i f(\boldsymbol{x}_i)\right\} + \lambda \|f\|^2_{\mathcal{H}_K}\right], \tag{4.4}$$

in which $\mathcal{H}_K$ is an RKHS generated by a positive definite kernel function K. The popular kernel functions include the Gaussian kernel and the polynomial kernel. By Mercer's theorem, kernel K has an eigen-expansion $K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \sum_{t=1}^{\infty} \gamma_t \varphi_t(\boldsymbol{x}_i)\varphi_t(\boldsymbol{x}_j)$ with $\gamma_t \ge 0$ and $\sum_{t=1}^{\infty} \gamma_t^2 < \infty$. The function f in the space $\mathcal{H}_K$ has an expansion in terms of eigenfunctions, $f(\boldsymbol{x}) = \sum_{t=1}^{\infty} c_t \varphi_t(\boldsymbol{x})$, where $\|f\|^2_{\mathcal{H}_K} \equiv \sum_{t=1}^{\infty} c_t^2/\gamma_t < \infty$.

By the representer theorem (Wahba, 1990), the solution of problem (4.4) has a finite form

$$\hat{f}(\boldsymbol{x}) = \sum_{i=1}^{n} \hat{\boldsymbol{\alpha}}_i K(\boldsymbol{x}, \boldsymbol{x}_i).$$

Then the reproducing property of the RKHS (Wahba, 1990) implies

$$\|\hat{f}\|^2_{\mathcal{H}_K} = \sum_{i=1}^{n}\sum_{i'=1}^{n} \hat{\alpha}_i \hat{\alpha}_{i'} K(\boldsymbol{x}_i, \boldsymbol{x}_{i'}),$$

and problem (4.4) becomes

$$\hat{\boldsymbol{\alpha}} = \underset{\boldsymbol{\alpha}\in\mathbb{R}^n}{\operatorname{argmin}} \left[\frac{1}{n}\sum_{i=1}^{n} \phi\left\{ y_i \boldsymbol{K}_i^\top \boldsymbol{\alpha} \right\} + \lambda \boldsymbol{\alpha}^\top \boldsymbol{K} \boldsymbol{\alpha} \right], \tag{4.5}$$

where $\boldsymbol{K}$ is an $n \times n$ matrix whose $(i, i')^{\text{th}}$ element is $K(\boldsymbol{x}_i, \boldsymbol{x}_{i'})$. Problem (4.5) can be efficiently solved based on the MM principle, as shown in Chapter 2, which is much faster than the second-order cone programming algorithm. The reproducing property of the RKHS largely facilitates the computation algorithms, as the implicit and infinite-dimensional problem (4.4) reduces to an explicit and finite-dimensional problem (4.5). The explicit feature map of the RKHS induced by the Guassian kernel has been studied in Steinwart et al. (2006).

4.3.2 Our proposal

The empirical loss minimization formulation of the original DWD is the first step towards the multicategory DWD. In the literature many efforts have been devoted to the multiclass generalization of the binary large margin classifier that can be formulated as an empirical loss minimization problem. For example, the multicategory SVM (Lee et al., 2004), the multicategory ψ-learning (Liu and Shen, 2006), and so on. Here, we take a different approach from the existing multicategory large margin classifiers in the literature. Specifically, we take advantage of the concept of *margin vector*, which is introduced by Zou et al. (2008) and is conceptually identical to the binary margin. In binary classification, the margin is defined as yf, which assigns margin $f(\boldsymbol{x}_i)$ to a data point $(\boldsymbol{x}_i, y_i)$ from positive class and assigns margin $-f(\boldsymbol{x}_i)$ to datum from negative class. The binary margin definition explicitly uses the special $1, -1$ coding of the class label. For a k class problem, a margin vector has the form of $\boldsymbol{f} = (f_1, \ldots, f_k)^\top$ with a sum-to-zero constraint $\sum_{j=1}^{k} f_j = 0$. Data point $(\boldsymbol{x}_i, y_i)$ belonging to class y_i has margin $f_{y_i}(\boldsymbol{x}_i)$, where $y_i \in \{1, 2, \ldots, k\}$. When

$k = 2$, by the sum-to-zero constraint we have $f_1(\boldsymbol{x}_i) = -f_2(\boldsymbol{x}_i)$. Thus, when we use $1, -1$ to code the classes 1 and 2, we have $f = f_1$ and the margin for $(\boldsymbol{x}_i, y_i)$ is $y_i f(\boldsymbol{x}_i)$, which is the definition of the margin.

Now we replace the margin $y_i f(\boldsymbol{x}_i)$ with the margin vector $f_{y_i}(\boldsymbol{x}_i)$ in problem (4.4) and end up with the formulation

$$\hat{\boldsymbol{f}} = \underset{f_j \in \mathcal{H}_K}{\operatorname{argmin}} \left[\frac{1}{n} \sum_{i=1}^{n} \phi \left\{ f_{y_i}(\boldsymbol{x}_i) \right\} + \lambda \sum_{j=1}^{k} ||f_j||^2_{\mathcal{H}_K} \right], \text{ subject to } \sum_{j=1}^{k} f_j = 0, \tag{4.6}$$

where ϕ is the DWD loss (4.3) and $\mathcal{H}_K$ is an RKHS generated by a positive definite kernel K. The multicategory DWD classifier is $\hat{y} = \operatorname{argmax}_{j \in \{1,2,\ldots,k\}} \hat{f}_j(\boldsymbol{x})$. For the actual multiclass classification problem with $k \geq 3$, the formulation (4.6) is fundamentally different from the binary case in problem (4.4) in terms of computational and theoretical treatments. Thus, multicategory kernel DWD is not a trivial extension of the binary kernel DWD. The computation of $\hat{\boldsymbol{f}}$ is discussed in Section 4, and its competitive performance is demonstrated in Section 5.

To appreciate the formulation (4.6), let us consider the ideal case when n is infinity and $\lambda = 0$. Define $p_j(\boldsymbol{x}) = P(y = j|\boldsymbol{x})$, $j \in \{1, \ldots, k\}$. Note that $\sum_{i=1}^{n} \phi\{f_{y_i}(\boldsymbol{x}_i\}/n$ becomes $\sum_{j=1}^{k} \phi\{f_j(\boldsymbol{x})\} p(y = j|\boldsymbol{x})$. Thus, the problem (4.6) becomes

$$\boldsymbol{f}^\star(\boldsymbol{x}) = \underset{\boldsymbol{f}}{\operatorname{argmin}} \left[\sum_{j=1}^{k} \phi\{f_j(\boldsymbol{x})\} p_j(\boldsymbol{x}) \right], \text{ subject to } \sum_{j=1}^{k} f_j(\boldsymbol{x}) = 0. \tag{4.7}$$

The population multicategory DWD classifier is $\hat{y} = \operatorname{argmax}_{j \in \{1,2,\ldots,k\}} f_j^\star(\boldsymbol{x})$.

Conceptually speaking, the population multicategory DWD classifier is the target of the proposed multicategory DWD classifier $\hat{\boldsymbol{f}}$. In the next theorem, we show that the population multicategory DWD classifier is actually the Bayes rule, which indicates that the proposed multicategory DWD classifier estimates the right target for the multiclass classification problem. Such a property is called Fisher consistency (Lin, 2004).

Theorem 4.1

(Multicategory Fisher Consistency) Assume that for each x (or with measure one) there is

a most likely label $j^\star$ such that $p_{j^\star}(\boldsymbol{x}) > p_j(\boldsymbol{x})\ \forall j \neq j^\star$, and there is the least possible label j_* such that $p_j(\boldsymbol{x}) > p_{j_*}(\boldsymbol{x})\ \forall j \neq j_*$. Then the solution of the problem (4.7) is given by

$$f_j^\star(\boldsymbol{x}) = \begin{cases} \frac{1}{2}\sqrt{\frac{p_j(\boldsymbol{x})}{p_{j_\star}(\boldsymbol{x})}}, & j \neq j_*, \\ -\frac{1}{2}\sum_{l \neq j_*}\sqrt{\frac{p_l(\boldsymbol{x})}{p_{j_\star}(\boldsymbol{x})}}, & j = j_*. \end{cases} \tag{4.8}$$

Consequently, Theorem (4.8) indicates that $\operatorname{argmax}_{j\in\{1,2,\ldots,k\}} f_j^\star(\boldsymbol{x}) = \operatorname{argmax}_{j\in\{1,2,\ldots,k\}} p_j(\boldsymbol{x})$, i.e., the population multicategory DWD is identical to the Bayes rule. □

Huang et al. (2013) also proved the Fisher consistency of their multicategory DWD that is based on pairwise differences in discriminant functions. However, their method only considered the linear DWD but not the more flexible kernel DWD. Based on its meaning, Fisher consistency is much more relevant when the classifier can be flexible and non-linear. In Section 5 the multicategory linear DWD is shown to be inconsistent in some simulation examples.

Although ((4.6)) is defined as a functional optimization problem in a possibly infinite dimensional functional space, the nice reproducing property of RKHS makes the computation of $\hat{\boldsymbol{f}}$ in problem (4.6) to be carried out in a finite-dimensional vector space.

Theorem 4.2

(Multicategory representer theorem) If $\mathcal{H}_K$ is generated by a positive definite kernel function K, then the solution of (4.6), $\hat{\boldsymbol{f}} = (\hat{f}_1, \ldots, \hat{f}_k)$, has a finite form,

$$\hat{f}_j(\boldsymbol{x}) = \sum_{i=1}^{n} \hat{\alpha}_{ij} K(\boldsymbol{x}, \boldsymbol{x}_i),\ j = 1, \ldots, k, \qquad \square$$

and

$$\sum_{j=1}^{k} \hat{\alpha}_{ij} = 0, \quad \forall i = 1, \ldots, n.$$

Define $\boldsymbol{K}$ to be the kernel matrix whose (i, i')th element is $K(\boldsymbol{x}_i, \boldsymbol{x}_{i'})$ and let $\boldsymbol{K}_i$ be the ith column vector. For each class j, Theorem 4.2 implies that there exists $\hat{\boldsymbol{\alpha}}_j$ such that

$\hat{f}_j(\boldsymbol{x}_i) = \boldsymbol{K}_i^\top \boldsymbol{\alpha}_j$, where $\hat{\boldsymbol{\alpha}}_j = (\hat{\alpha}_{1j}, \ldots, \hat{\alpha}_{nj})^\top$. Now we only need to compute $\hat{\boldsymbol{\alpha}}_j$ for $j = 1, \ldots, k$.

By the reproducing property, it can be further obtained that

$$\|\hat{f}_j\|^2_{\mathcal{H}_K} = \sum_{i=1}^{n}\sum_{i'=1}^{n} \hat{\alpha}_{ij}\hat{\alpha}_{i'j} K(\boldsymbol{x}_i, \boldsymbol{x}_{i'}) = \hat{\boldsymbol{\alpha}}_j^\top \boldsymbol{K} \hat{\boldsymbol{\alpha}}_j.$$

Note that $f_{y_i}(\boldsymbol{x}_i) = \boldsymbol{K}_i^\top \boldsymbol{\alpha}_{y_i}$. Then, we can rephrase the optimization problem (4.6) as follows

$$\min_{\boldsymbol{\alpha}_j \in \mathbb{R}^n} \left[\frac{1}{n} \sum_{i=1}^{n} \phi\left\{ \boldsymbol{K}_i^\top \boldsymbol{\alpha}_{y_i} \right\} + \lambda \sum_{j=1}^{k} \boldsymbol{\alpha}_j^\top \boldsymbol{K} \boldsymbol{\alpha}_j \right], \text{ subject to } \sum_{j=1}^{k} \boldsymbol{\alpha}_j = \boldsymbol{0}. \tag{4.9}$$

In Section 4.4, we shall derive a efficient algorithm to solve the problem (4.9).

4.3.3 Related methods

To connect our method with other simultaneous multiclass large-margin classifiers in the literature, we formulate the loss of our proposal (4.7) as

$$\min_{\boldsymbol{f}} E_{\boldsymbol{x}y} \phi\{f_y(\boldsymbol{x})\}, \text{ subject to } \sum_{j=1}^{k} f_j(\boldsymbol{x}) = 0.$$

Vapnik (1998), Bredensteiner and Bennett (1999), and Weston and Watkins (1999) proposed multiclass SVMs, all of which, as shown by Guermeur (2002), can be written equivalently as

$$\min_{\boldsymbol{f}} E_{\boldsymbol{x}y} \sum_{j \neq y} \left[1 - (f_y(\boldsymbol{x}) - f_j(\boldsymbol{x}))\right]_+, \tag{4.10}$$

where $[w] = \max(w, 0)$. Crammer and Singer (2001) presented another multiclass SVM as

$$\min_{\boldsymbol{f}} E_{\boldsymbol{x}y} \left[1 - \min_{j \neq y}(f_y(\boldsymbol{x}) - f_j(\boldsymbol{x}))\right]_+. \tag{4.11}$$

Lee et al. (2004) developed multiclass SVM as

$$\min_{\boldsymbol{f}} E_{\boldsymbol{x}y} \sum_{j \neq y} [1 + f_j(\boldsymbol{x})]_+ , \text{ subject to } \sum_{j=1}^{k} f_j(\boldsymbol{x}) = 0, \tag{4.12}$$

and they showed that their proposal is Fisher consistent but the methods (4.10) and (4.11) are not. Liu and Shen (2006) introduced multicategory ψ-learning

$$\min_{\boldsymbol{f}} E_{\boldsymbol{x}y} \left[1 - \left(\min_{j \neq y} (f_y(\boldsymbol{x}) - f_j(\boldsymbol{x})) \right)_+ \right]_+$$

by replacing the convex SVM hinge loss with a non-convex ψ-loss. Liu and Yuan (2011) proposed reinforced multiclass SVM, on the basis of the linear combination of the SVM hinge loss and the loss function in Lee et al. (2004):

$$\min_{\boldsymbol{f}} E_{\boldsymbol{x}y} \sum_{j \neq y} [(1-\gamma)(1 - f_y(\boldsymbol{x}))_+ + \gamma(1 + f_j(\boldsymbol{x}))_+] , \text{ subject to } \sum_{j=1}^{k} f_j(\boldsymbol{x}) = 0, \tag{4.13}$$

which is shown to enjoy the Fisher consistency when $\gamma \in [1/2, 1]$.

Among the aforementioned approaches, the sum-to-zero constraint $\sum_{j=1}^{k} f_j(\boldsymbol{x}) = 0$ is enforced in the methods (4.12) and (4.13) and can be also imposed in others to ensure the uniqueness of the optimal solution. To avoid the explicit sum-to-zero constraint of those methods, Zhang and Liu (2014) proposed a novel angle-based approach, fitting a model based on the angles between data and each vertex vector of a k-simplex. We take as an example the reinforced angle-based multiclass SVM (RAMSVM, Zhang et al., 2016), which is developed by applying the angle-based approach to the reinforced multiclass SVM (4.13). Specifically, the angle-based approach first finds a k-simplex that consists of k unit-norm vertices $\{\boldsymbol{W}_j\}_{j=1}^{k} \in \mathbb{R}^{k-1}$ such that the angles between the pairs $(\boldsymbol{W}_j, \boldsymbol{W}_{j'})$ are the same. The model is then fitted by replacing each functional margin $f_j(\boldsymbol{x})$ in (4.13) by $\langle \boldsymbol{f}, \boldsymbol{W}_j \rangle$:

$$\min_{\boldsymbol{f}} E_{\boldsymbol{x}y} \sum_{j \neq y} \left[\frac{1}{2}(1 - \langle \boldsymbol{f}(\boldsymbol{x}), \boldsymbol{W}_y \rangle)_+ + \frac{1}{2}(1 + \langle \boldsymbol{f}(\boldsymbol{x}), \boldsymbol{W}_j \rangle)_+ \right].$$

The prediction is made according to $\hat{y} = \text{argmax}_j \langle \boldsymbol{f}(x), \boldsymbol{W}_j \rangle$. Since $\sum_{j=1}^{k} \langle \boldsymbol{f}(\boldsymbol{x}), \boldsymbol{W}_j \rangle = 0$ always holds, the sum-to-zero constraint is dismissed. Other applications of the angle-based approaches are seen in Sun et al. (2017); Zhang et al. (2017); Fu et al. (2018); Liu et al. (2018). Compared with the angle-based method, our method has the explicit sum-to-zero constraint. As will be shown in Section 4.4, our algorithm handles such constraint quite naturally and efficiently.

4.4 Computation Algorithm

multicategory kernel DWD problem (4.9) is more sophisticated than the binary kernel DWD problem (4.4) due to the sum-to-zero constraint. In this section, we derive an accelerated projected gradient descent (PGD) algorithm to solve problem (4.9).

4.4.1 Projected gradient descent algorithm

We first derive the projected gradient descent algorithm and then derive its accelerated version.

Notation. $\boldsymbol{A} \otimes \boldsymbol{B}$ denotes the *Kronecker product* of an $m \times n$ matrix $\boldsymbol{A}$ and a $p \times q$ matrix $\boldsymbol{B}$ is the $mp \times nq$ matrix

$$\boldsymbol{A} \otimes \boldsymbol{B} = \begin{pmatrix} a_{11}\boldsymbol{B} & a_{12}\boldsymbol{B} & \dots & a_{1n}\boldsymbol{B} \\ a_{21}\boldsymbol{B} & a_{22}\boldsymbol{B} & \dots & a_{2n}\boldsymbol{B} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1}\boldsymbol{B} & a_{m2}\boldsymbol{B} & \dots & a_{mn}\boldsymbol{B} \end{pmatrix}.$$

The *vectorization* of a $m \times n$ matrix $\boldsymbol{A}$ converts the matrix into a mn-column vector by stacking the first, second, ..., n^{th} columns $\boldsymbol{a}_1, \boldsymbol{a}_2, \dots, \boldsymbol{a}_n$ of $\boldsymbol{A}$ one under the other:

$$\text{vec}(\boldsymbol{A}) = (\boldsymbol{a}_1^\top, \boldsymbol{a}_2^\top, \dots, \boldsymbol{a}_n^\top)^\top.$$

Consider a constrained minimization problem over a convex set $\mathcal{A}$:

$$\min F(\boldsymbol{\alpha}), \text{ subject to } \boldsymbol{\alpha} \in \mathcal{A},$$

where F is a continuously differentiable and strongly convex function. For $t = 0, 1, 2, \ldots$, the PGD algorithm updates

$$\begin{aligned} \boldsymbol{\alpha}^{(t+1)} &= \text{proj}_{\mathcal{A}} \left(\boldsymbol{\alpha}^{(t)} - d_t \nabla F(\boldsymbol{\alpha}^{(t)}) \right) \\ &\equiv \underset{\boldsymbol{\alpha} \in \mathcal{A}}{\text{argmin}} \left\| \boldsymbol{\alpha} - \left(\boldsymbol{\alpha}^{(t)} - d_t \nabla F(\boldsymbol{\alpha}^{(t)}) \right) \right\|^2, \end{aligned} \tag{4.14}$$

where d_t is a step size that we shall determine. If the algorithm converges to $\boldsymbol{\alpha}^\star$ such that

$$\boldsymbol{\alpha}^\star = \text{proj}_{\mathcal{A}} \left(\boldsymbol{\alpha}^\star - d_t \nabla F(\boldsymbol{\alpha}^\star) \right),$$

then one can observe that $\boldsymbol{\alpha}^\star \in \mathcal{A}$ and $\nabla F(\boldsymbol{\alpha}^\star) = \mathbf{0}$ by differentiating equation (4.14) in terms of $\boldsymbol{\alpha}$. Hence $\boldsymbol{\alpha}^\star$ is a global minimizer of F on the set $\mathcal{A}$.

We next apply the PGD algorithm to solve the optimization problem (4.9). Suppose $\boldsymbol{A}$ is an $n \times k$ matrix whose j^{th} column is $\boldsymbol{\alpha}_j$, then $\boldsymbol{A}\boldsymbol{e}_j = \boldsymbol{\alpha}_j$, where $\boldsymbol{e}_j$ a k-vector whose elements are 0 except that the j^{th} element is 1. We observe that the constraint $\sum \boldsymbol{\alpha}_j = 0$ in problem (4.9) amounts to $\boldsymbol{A}\mathbf{1}_k = \mathbf{0}$, where $\mathbf{1}_k$ is the k-vector of 1's. Let $\boldsymbol{\alpha} = \text{vec}(\boldsymbol{A})$, we have

$$\begin{aligned} \boldsymbol{A}\boldsymbol{e}_j &= \text{vec}(\boldsymbol{A}\boldsymbol{e}_j) = (\boldsymbol{e}_j^\top \otimes \boldsymbol{I}_n)\boldsymbol{\alpha}, \\ \boldsymbol{A}\mathbf{1}_k &= \text{vec}(\boldsymbol{A}\mathbf{1}_k) = (\mathbf{1}_k^\top \otimes \boldsymbol{I}_n)\boldsymbol{\alpha}. \end{aligned}$$

Accordingly, the optimization problem (4.9) can be written as

$$\min_{\boldsymbol{\alpha} \in \mathbb{R}^{nk}} F(\boldsymbol{\alpha}) = \min_{\boldsymbol{\alpha} \in \mathbb{R}^{nk}} \left[\frac{1}{n} \sum_{i=1}^{n} \phi \left\{ \boldsymbol{K}_i^\top (\boldsymbol{e}_{y_i}^\top \otimes \boldsymbol{I}_n)\boldsymbol{\alpha} \right\} + \lambda \sum_{j=1}^{k} \boldsymbol{\alpha}^\top (\boldsymbol{e}_j \otimes \boldsymbol{I}_n) \boldsymbol{K} (\boldsymbol{e}_j^\top \otimes \boldsymbol{I}_n)\boldsymbol{\alpha} \right], \quad \text{subject to } (\mathbf{1}_k^\top \otimes \boldsymbol{I}_n)\boldsymbol{\alpha} = \mathbf{0}. \tag{4.15}$$

Let $\boldsymbol{B} = \mathbf{1}_k^\top \otimes \boldsymbol{I}_n$. By the PGD algorithm introduced in equation (4.14), problem (4.15)

can be solved as

$$\boldsymbol{\alpha}^{(t+1)} = \underset{\{\boldsymbol{\alpha}:\boldsymbol{B}\boldsymbol{\alpha}=\mathbf{0}\}}{\operatorname{argmin}} \|\boldsymbol{\alpha}-\tilde{\boldsymbol{\alpha}}\|^2 = \tilde{\boldsymbol{\alpha}} - \boldsymbol{B}^\top(\boldsymbol{B}\boldsymbol{B}^\top)^{-1}\boldsymbol{B}\tilde{\boldsymbol{\alpha}} = \tilde{\boldsymbol{\alpha}} - ((\mathbf{1}_k\mathbf{1}_k^\top)\otimes\boldsymbol{I}_n)\tilde{\boldsymbol{\alpha}},$$

where $\tilde{\boldsymbol{\alpha}} = \boldsymbol{\alpha}^{(t)} - d_t\nabla F(\boldsymbol{\alpha}^{(t)})$, and

$$\nabla F(\boldsymbol{\alpha}^{(t)}) = \frac{1}{n}\sum_{i=1}^{n}\phi'\left\{\boldsymbol{K}_i^\top(\boldsymbol{e}_{y_i}^\top\otimes\boldsymbol{I}_n)\boldsymbol{\alpha}^{(t)}\right\}(\boldsymbol{e}_{y_i}\otimes\boldsymbol{I}_n)\boldsymbol{K}_i + 2\lambda(\boldsymbol{I}_k\otimes\boldsymbol{K})\boldsymbol{\alpha}^{(t)}. \quad (4.16)$$

We use a linear search method to determine the step size d_t. Specifically, at each iteration t, with a pre-defined constant $\eta < 1$, we find the smallest non-negative integer b such that $d_t = \eta^b d_{t-1}$ and

$$F(\boldsymbol{\alpha}^+) \le F(\boldsymbol{\alpha}^{(t)}) + \nabla F(\boldsymbol{\alpha}^{(t)})(\boldsymbol{\alpha}^+ - \boldsymbol{\alpha}^{(t)}) + \frac{1}{2d_t}||\boldsymbol{\alpha}^+ - \boldsymbol{\alpha}^{(t)}||_2^2, \quad (4.17)$$

where $\tilde{\boldsymbol{\alpha}} = \boldsymbol{\alpha}^{(t)} - d_t\nabla F(\boldsymbol{\alpha}^{(t)})$ and $\boldsymbol{\alpha}^+ = \tilde{\boldsymbol{\alpha}} - \boldsymbol{B}^\top(\boldsymbol{B}\boldsymbol{B}^\top)^{-1}\boldsymbol{B}\tilde{\boldsymbol{\alpha}}$. Then we set $F(\boldsymbol{\alpha}^{(t+1)}) = F(\boldsymbol{\alpha}^+)$.

Algorithm 2 summarizes the details of the PGD algorithm. The rate of convergence is given in proposition 4.1.

Proposition 4.1

Let $\boldsymbol{\alpha}^{(t)}$ be the sequence generated by Algorithm 2 and $\boldsymbol{\alpha}^\star$ is the global minimizer of problem (4.15). Then for any $t \ge 1$,

$$F(\boldsymbol{\alpha}^{(t)}) - F(\boldsymbol{\alpha}^\star) \le \frac{c_1}{t}\|\boldsymbol{\alpha}^{(0)} - \boldsymbol{\alpha}^\star\|^2,$$

in which $c_1 = 2\eta\tilde{\sigma}/n + \eta\lambda\sigma$, $\tilde{\sigma} = \max_j \tilde{\sigma}_j$ where each $\tilde{\sigma}_j$ is the largest eigenvalue of $\sum_{\{i:y_i=j\}}\boldsymbol{K}_i\boldsymbol{K}_i^\top$, and σ is the largest eigenvalue of $\boldsymbol{K}$.

Proposition 4.1 implies that $\mathcal{O}(1/\varepsilon)$ iterations are needed to reach $F(\boldsymbol{\alpha}^{(t)}) - F(\boldsymbol{\alpha}^\star) < \varepsilon$.

Algorithm 2 Projected Gradient Descent Algorithm for Multicategory Kernel DWD (4.15)

Initialize $\boldsymbol{\alpha}^{(0)}$, step size $d = 1$, $\eta = 0.5$, $\boldsymbol{B} = \mathbf{1}_k^\top \otimes \boldsymbol{I}_n$, and $t = 0$
repeat
  Compute $\nabla F(\boldsymbol{\alpha}^{(t)})$ as in (4.16)
  repeat
    Set $d = d\eta$
    Compute $\tilde{\boldsymbol{\alpha}} = \boldsymbol{\alpha}^{(t)} - d\nabla F(\boldsymbol{\alpha}^{(t)})$ and $\boldsymbol{\alpha}^{+} = \tilde{\boldsymbol{\alpha}} - \boldsymbol{B}^\top(\boldsymbol{B}\boldsymbol{B}^\top)^{-1}\boldsymbol{B}\tilde{\boldsymbol{\alpha}}$
  until the condition (4.17) is satisfied
  Set $\boldsymbol{\alpha}^{(t+1)} = \boldsymbol{\alpha}^{+}$
  Set $t = t + 1$
until a convergence condition is met

4.4.2 PGD algorithm with the Nesterov's acceleration

In this section, we further develop an accelerated PGD algorithm by employing the Nesterov's acceleration (Beck and Teboulle, 2009; Nesterov, 2013). The improved algorithm has a much faster rate of converge.

The accelerated PGD algorithm generates a number sequence δ_t such that

$$\delta_{t+1} = \frac{1 + \sqrt{1 + 4\delta_t^2}}{2},$$

and a sequence of $\boldsymbol{\beta}^{(t)}$ along with $\alpha^{(t)}$. With $\boldsymbol{\beta}^{(0)} = \boldsymbol{\alpha}^{(0)}$ initialized and $\delta_1 = 1$, the algorithm updates

$$\boldsymbol{\alpha}^{(t+1)} = \underset{\{\boldsymbol{\alpha}:\boldsymbol{B}\boldsymbol{\alpha}=\mathbf{0}\}}{\operatorname{argmin}} \left\| \boldsymbol{\alpha} - \left(\boldsymbol{\beta}^{(t)} - d_t \nabla F(\boldsymbol{\beta}^{(t)}) \right) \right\|^2,$$
$$\boldsymbol{\beta}^{(t+1)} = \boldsymbol{\alpha}^{(t+1)} + \frac{\delta_t - 1}{\delta_{t+1}} (\boldsymbol{\alpha}^{(t)} - \boldsymbol{\alpha}^{(t-1)}).$$

The accelerated PGD algorithm is summarized in Algorithm 3. The rate of convergence is $\mathcal{O}(1/t^2)$, as presented in proposition 4.2.

Proposition 4.2

Let $\boldsymbol{\alpha}^{(t)}$ be the sequence generated by Algorithm 3 and $\boldsymbol{\alpha}^\star$ is the global minimizer of

Algorithm 3 Accelerated Projected Gradient Descent for Multicategory Kernel DWD (4.15)

Initialize $\boldsymbol{\alpha}^{(0)} = \boldsymbol{\beta}^{(1)}$, step size $d = 1$, $\eta = 0.5$, $\boldsymbol{B} = \mathbf{1}_k^\top \otimes \boldsymbol{I}_n$, $\delta_1 = 1$, and $t = 1$
repeat
  Compute $\nabla F(\boldsymbol{\beta}^{(t)})$ as in (4.16)
  repeat
    Set $d = d\eta$
    Compute $\tilde{\boldsymbol{\alpha}} = \boldsymbol{\beta}^{(t)} - d\nabla F(\boldsymbol{\beta}^{(t)})$ and $\boldsymbol{\alpha}^+ = \tilde{\boldsymbol{\beta}} - \boldsymbol{B}^\top(\boldsymbol{B}\boldsymbol{B}^\top)^{-1}\boldsymbol{B}\tilde{\boldsymbol{\beta}}$
  until the condition (4.17) is satisfied
  Set $\boldsymbol{\alpha}^{(t)} = \boldsymbol{\alpha}^+$
  Compute $\delta_{t+1} = \frac{1+\sqrt{1+4\delta_t^2}}{2}$.
  Set $\boldsymbol{\beta}^{(t+1)} = \boldsymbol{\alpha}^{(t)} + \frac{\delta_t - 1}{\delta_{t+1}}(\boldsymbol{\alpha}^{(t)} - \boldsymbol{\alpha}^{(t-1)})$
  Set $t = t + 1$
until a convergence condition is met

problem (4.15). Then for any $t \geq 1$,

$$F(\boldsymbol{\alpha}^{(t)}) - F(\boldsymbol{\alpha}^\star) \leq \frac{c_2}{t^2}\|\boldsymbol{\alpha}^{(0)} - \boldsymbol{\alpha}^\star\|^2,$$

in which $c_2 = 8\eta\tilde{\sigma}/n + 4\eta\lambda\sigma$, $\tilde{\sigma} = \max_j \tilde{\sigma}_j$ where each $\tilde{\sigma}_j$ is the largest eigenvalue of $\sum_{\{i:y_i=j\}} \boldsymbol{K}_i\boldsymbol{K}_i^\top$, and σ is the largest eigenvalue of $\boldsymbol{K}$.

Proposition 4.2 implies that, by the Nesterov's acceleration, $\mathcal{O}(1/\sqrt{\varepsilon})$ iterations are needed to reach $F(\boldsymbol{\alpha}^{(t)}) - F(\boldsymbol{\alpha}^\star) < \varepsilon$.

4.4.3 Implementation

We have implemented the accelerated PGD algorithm for solving multicategory kernel DWD in an R package `mdwd`, which is available upon request. Users can choose the kernel function and use cross-validation to select the regularization parameter λ.

4.5 Numerical Studies

In this section, we use simulations and benchmark data applications to compare our multicategory kernel DWD with the multicategory DWD proposed by Huang et al. (2013) and

implemented in the R package `DWD` (Huang et al., 2012). We also compare our method with off-the-shelf multicategory classifiers in R: multiclass kernel SVM in the R package `SMSVM` (Lee et al., 2004), reinforced angle-based multiclass SVM in the R package `RAMSVM` (Zhang et al., 2016), random forest in the R package `randomForest` (Liaw and Wiener, 2002), gradient boosting machines in the R package `gbm` (Ridgeway, 2017), and k-nearest neighbors in the R package `class` (Venables and Ripley, 2002). The Gaussian kernels are employed for `mdwd`, `SMSVM`, and `RAMSVM`, and the tuning parameters are estimated by five-fold cross-validations. For the k-nearest neighbors, k is chosen from $(3, 4, \ldots, 9)$.

4.5.1 Simulations

We followed the simulation setting that was used in Section 4.1. Example 1 is a three-category classification, i.e., $k = 3$, and the dimension $p = 5$. For each class $j = 1, 2, 3$, we independently generated three centers $\boldsymbol{\mu}_{j,1}, \boldsymbol{\mu}_{j,2}$, and $\boldsymbol{\mu}_{j,3}$ following $N(\mathbf{0}, \boldsymbol{I}_{3\times 3})$. We generated each data point $(\boldsymbol{x}_i, y_i)$ by first assigning a class label to y_i with equal conditional class probabilities and then randomly having a center $\boldsymbol{\mu}_{y_i,l}$ where $l = 1, 2, 3$ to draw $\boldsymbol{x}_i$ from $N(\boldsymbol{\mu}_{y_i,l}, \tau^2 \boldsymbol{I}_{p\times p})$. In example 1, we set $\tau = 0.5$ and the corresponding Bayes error rate is 6.39%. We assembled the training data with the sample size varying over $100, 200, 400, 600$, and 800. We trained and tuned each method on the training data, and we investigated the prediction error on a test set consisting of 10000 independently generated observations.

Table 4.1 exhibits the mean misclassification rates and the standard errors, averaged by 100 replicates. For example 1, we observe that multicategory kernel DWD delivers the least prediction error among the seven multicategory classifiers. The SVM and k-nearest neighbors have slightly worse performance than our proposal. We discover that the prediction error of our method decreases and also approaches the Bayes error as the sample size increases, which indicates that multicategory kernel DWD loss has a right target function and is thus Fisher consistent. Multicategory linear DWD does not work well and is far from the Bayes rate.

Examples 2, 3, and 4 adopt the same simulation settings as example 1, except that

example 2 sets $k = 3$, $p = 20$, and $\tau = 1.5$ yielding the Bayes error of 9.39%, example 3 has $k = 3$, $p = 2$, $\tau = 0.4$ and the Bayes error of 25.27%, and example 4 contains four categories in the response, $k = 4$, and it has the Bayes error of 8.78%. As seen in Table 4.1, multicategory kernel DWD has the best prediction accuracy in all those three examples, two variants of SVM have slightly worse but very competitive behaviors, whereas multicategory linear DWD has the worst accuracy in general.

In example 5, we generated each class center $\boldsymbol{\mu}_y$ from the standard normal distribution, and we then drew each data point from $N(\boldsymbol{\mu}_{y_i}, 2^2\boldsymbol{I}_{20\times 20})$. The Bayes error is 11.45%, and the true decision boundaries between the classes are actually linear. From Table 4.1, we see that the classification error of multicategory linear DWD is the lowest and approaches the Bayes error. The performance of our proposal and the angle-based method follows intimately as well.

To sum up, the simulation examples have clearly conveyed the following messages:

- the proposed multicategory kernel DWD works much better than the multicategory linear DWD when the underlying Bayes rule is nonlinear;
- the proposed multicategory kernel DWD delivers lower classification error that approaches the Bayes error when the training size increases;
- the proposed multicategory kernel DWD has very competitive performance against other popular off-the-shelf multiclass classifiers, although none dominates the rest.

4.5.2 Benchmark data applications

We examined the performance of multicategory kernel DWD on eight benchmark data sets that were downloaded at University of California at Irvine Machine Learning Repository (Dua, D. and Karra Taniskidou, E., 2017). We compared our method with the same methods that were used in Section 4.5.1. In the tables of this section, the total sample size of each data, the number of categories in the response, and the dimension are denoted by N, k, and p, respectively. For each data set, we held out $n_{\text{test}} = N - 800$ observations as test data, and we randomly selected n_{train} observations as training data to train and tune each method, where the sample size of the training data n_{train} varied over 100, 200, 300, 400,

600 and 800. Averaged over 40 independent random splits, the misclassification error and computation time are summarized in Table 4.2, Table 4.3, and Table 4.4. The computation time in Table 4.4 includes fitting the models and tuning the parameters. We ranked the error and time of each method in Table 4.5 and Table 4.6.

From the prediction error in Table 4.2 and Table 4.3 and the ranks in Table 4.5, we find that our proposal has the lowest prediction error on two benchmark data, `vowel` and `waveform`, as well as the second lowest error on three examples, `abalone`, `pendigits`, and `satimage`. Multicategory linear DWD suffers from the worst accuracy on five data sets so it appears inadequate for these real applications. From Table 4.5, we see that the overall performance of our proposal on these benchmark data outperforms the SVM and the angle-based approach but it is worse than random forest. In terms of computation time, Table 4.4 and Table 4.6 show that our implementation `mdwd` is the fastest among the four large-margin classifiers. The algorithms implemented in the packages `SMSVM` and `DWD` did not even converge within ten hours in several cases. We discover that random forest and gradient boosting are faster than these large-margin classifiers. We further notice the computation time of our proposal is relatively unaffected when there many categories in the response, for example, the pendigits and vowel data; nonetheless, the computing speed of other large-margin classifiers is dramatically degraded as the number k increases. Although the prediction accuracy of k-nearest neighbors is among the worse, it runs the fastest in all examples.

According to the performance on the eight benchmark data, it is clear that our multicategory kernel DWD is much better than multicategory linear DWD by the `DWD` package and also highly competitive with other popular classifiers. The random forest has the overall best performance. We also implemented the polynomial kernel and found its performance is worse than that by the Gaussian kernel in these examples. For sake of space we do not include the result of DWD with polynomial kernel here.

Table 4.1: Prediction error (%) on mixture Gaussian simulation examples. Our multicategory kernel DWD (denoted by WZ) are compared with DWD (denoted by HLD (Huang et al., 2013)), multiclass kernel SVM (R package `SMSVM`), reinforcement angle-based multiclass SVM (R package `ramsvm`), random forest (R package `randomForest`), gradient boosting machines (R package `gbm`), and k-nearest neighbors (R package `class`). The results are averaged by 100 independent runs, and the standard error of the mean prediction error is given in parentheses. For each case, the method incurring the lowest error is marked by italics.

n	*Prediction error (%) for the following methods:*													
	DWD WZ		*DWD HLD*		*Multiclass SVM*		*Angle-based MSVM*		*Random forest*		*Gradient boosting*		*k-nearest neighbors*	
Example 1: $k = 3, p = 5$, Bayes error: 6.39														
100	*10.06*	(0.48)	25.60	(0.94)	11.68	(0.52)	12.00	(0.52)	14.08	(0.51)	18.86	(0.58)	11.34	(0.54)
200	*8.92*	(0.44)	24.79	(0.95)	9.62	(0.45)	9.93	(0.48)	11.35	(0.45)	16.58	(0.56)	9.38	(0.48)
400	*7.88*	(0.41)	24.32	(0.92)	8.46	(0.41)	8.99	(0.44)	9.75	(0.41)	15.37	(0.54)	8.51	(0.45)
600	*7.59*	(0.39)	24.27	(0.92)	8.05	(0.39)	8.87	(0.43)	9.08	(0.40)	15.01	(0.53)	8.24	(0.43)
800	*7.38*	(0.38)	24.21	(0.93)	7.78	(0.39)	8.76	(0.43)	8.69	(0.40)	14.80	(0.53)	7.94	(0.41)
Example 2: $k = 3, p = 20$, Bayes error: 9.39														
100	*22.40*	(0.40)	27.30	(0.44)	23.07	(0.38)	24.51	(0.42)	27.58	(0.40)	30.31	(0.41)	26.26	(0.43)
200	*18.08*	(0.32)	24.78	(0.40)	18.48	(0.29)	20.41	(0.37)	23.18	(0.31)	27.03	(0.36)	21.44	(0.38)
400	*14.91*	(0.28)	23.33	(0.40)	15.42	(0.26)	17.61	(0.33)	19.95	(0.30)	24.95	(0.33)	18.16	(0.34)
600	*13.82*	(0.27)	22.77	(0.38)	14.30	(0.26)	16.41	(0.30)	18.61	(0.28)	24.30	(0.33)	16.92	(0.32)
800	*13.09*	(0.25)	22.50	(0.38)	13.66	(0.24)	16.04	(0.32)	17.84	(0.27)	23.97	(0.32)	16.01	(0.30)
Example 3: $k = 3, p = 2$, Bayes error: 25.27														
100	*29.82*	(0.80)	42.24	(0.96)	30.18	(0.77)	30.93	(0.78)	31.96	(0.79)	33.80	(0.76)	30.87	(0.83)
200	*27.97*	(0.78)	41.75	(0.98)	28.32	(0.76)	29.23	(0.78)	30.52	(0.81)	31.53	(0.77)	29.29	(0.81)
400	*26.97*	(0.74)	41.46	(0.97)	27.12	(0.74)	27.90	(0.75)	29.58	(0.78)	30.63	(0.75)	28.36	(0.78)
600	*26.76*	(0.74)	41.49	(0.96)	26.81	(0.74)	27.80	(0.73)	29.45	(0.80)	30.29	(0.75)	28.28	(0.80)
800	*26.55*	(0.73)	41.41	(0.94)	26.61	(0.72)	27.67	(0.73)	29.32	(0.79)	29.99	(0.72)	28.21	(0.78)
Example 4: $k = 4, p = 5$, Bayes error: 8.78														
100	*15.04*	(0.52)	34.83	(0.91)	17.38	(0.55)	17.31	(0.52)	19.98	(0.52)	27.27	(0.61)	16.84	(0.54)
200	*12.16*	(0.44)	33.62	(0.89)	13.65	(0.49)	14.41	(0.50)	15.62	(0.46)	23.78	(0.56)	13.21	(0.48)
400	*10.86*	(0.39)	33.33	(0.92)	11.86	(0.43)	12.99	(0.46)	13.47	(0.40)	22.26	(0.54)	11.67	(0.42)
600	*10.50*	(0.39)	33.32	(0.91)	11.21	(0.40)	12.69	(0.43)	12.55	(0.39)	21.77	(0.52)	11.33	(0.42)
800	*10.16*	(0.38)	33.29	(0.91)	10.72	(0.38)	12.44	(0.44)	11.95	(0.39)	21.50	(0.54)	10.80	(0.40)
Example 5: $k = 3, p = 20$, Bayes error: 11.45 (linear decision boundary)														
100	16.39	(0.38)	*15.30*	(0.36)	17.33	(0.36)	16.82	(0.40)	19.34	(0.38)	20.11	(0.36)	22.71	(0.47)
200	14.18	(0.36)	*13.39*	(0.32)	14.95	(0.33)	14.07	(0.35)	17.13	(0.36)	17.50	(0.34)	19.85	(0.45)
400	13.18	(0.32)	*12.54*	(0.30)	13.74	(0.31)	12.91	(0.31)	15.60	(0.32)	16.04	(0.31)	17.92	(0.41)
600	12.77	(0.32)	*12.20*	(0.30)	13.39	(0.31)	12.59	(0.31)	15.11	(0.32)	15.71	(0.31)	17.19	(0.40)
800	12.54	(0.31)	*12.07*	(0.29)	12.98	(0.30)	12.34	(0.30)	14.90	(0.31)	15.60	(0.30)	16.81	(0.39)

Table 4.2: Prediction error (%) on benchmark data applications. Our multicategory kernel DWD (denoted by WZ) are compared with DWD (denoted by HLD (Huang et al., 2013)), multiclass kernel SVM (R package `SMSVM`), reinforcement angle-based multiclass SVM (R package `ramsvm`), random forest (R package `randomForest`), gradient boosting machines (R package `gbm`), and k-nearest neighbors (R package `class`). The results are averaged by 40 independent runs, and the standard error of the mean prediction error is given in parentheses. For each data, k and p are the number of classes and dimensions, and the total sample size is N. For each case, the method incurring the lowest error is marked by italics. Cases when the algorithm did not converge within ten hours are marked as "*".

n_{train}	*Prediction error (%) for the following methods:*													
	DWD WZ		*DWD HLD*		*Multiclass SVM*		*Angle-based MSVM*		*Random forest*		*Gradient boosting*		*k-nearest neighbors*	
`abalone`: $k = 3, p = 9, N = 4177, n_{\text{test}} = 3377$														
100	40.05	(0.36)	*39.55*	(0.21)	39.69	(0.35)	40.20	(0.37)	40.78	(0.29)	39.74	(0.26)	42.84	(0.26)
200	37.88	(0.18)	38.20	(0.17)	*37.80*	(0.21)	38.15	(0.30)	39.00	(0.19)	38.34	(0.20)	41.93	(0.28)
300	*37.02*	(0.21)	37.65	(0.16)	37.20	(0.20)	37.52	(0.32)	38.16	(0.15)	37.84	(0.18)	40.93	(0.19)
400	*36.35*	(0.18)	37.35	(0.14)	36.63	(0.21)	36.61	(0.25)	37.52	(0.15)	37.53	(0.19)	40.14	(0.21)
600	35.96	(0.11)	37.38	(0.12)	*35.64*	(0.13)	36.09	(0.19)	36.98	(0.12)	37.24	(0.15)	40.00	(0.15)
800	35.33	(0.11)	36.91	(0.10)	*35.21*	(0.09)	35.73	(0.12)	36.58	(0.09)	36.76	(0.10)	39.21	(0.16)
`covtype`: $k = 3, p = 10, N = 73631, n_{\text{test}} = 72831$														
100	25.25	(0.26)	28.09	(0.33)	23.21	(0.19)	23.69	(0.26)	*21.91*	(0.21)	22.83	(0.27)	27.90	(0.23)
200	22.87	(0.21)	27.09	(0.28)	22.62	(0.21)	22.00	(0.23)	*20.41*	(0.19)	21.62	(0.16)	25.23	(0.16)
300	21.33	(0.15)	27.00	(0.24)	21.44	(0.22)	20.88	(0.20)	*18.90*	(0.12)	21.00	(0.15)	23.87	(0.19)
400	20.61	(0.10)	26.55	(0.19)	20.44	(0.19)	20.31	(0.16)	*18.32*	(0.10)	20.56	(0.12)	22.80	(0.12)
600	19.44	(0.11)	26.48	(0.15)	19.52	(0.20)	19.94	(0.13)	*16.92*	(0.08)	20.12	(0.12)	21.58	(0.13)
800	18.50	(0.10)	26.13	(0.15)	18.79	(0.14)	19.73	(0.12)	*16.22*	(0.09)	19.92	(0.08)	20.67	(0.09)
`pendigits`: $k = 10, p = 16, N = 10990, n_{\text{test}} = 10190$														
100	10.03	(0.21)	21.02	(0.35)	11.41	(0.28)	*9.69*	(0.26)	13.53	(0.23)	25.18	(0.41)	13.91	(0.30)
200	6.48	(0.15)	19.47	(0.29)	7.45	(0.23)	*5.80*	(0.18)	8.59	(0.16)	18.66	(0.20)	9.08	(0.16)
300	4.84	(0.11)	*	(*)	5.14	(0.15)	*4.41*	(0.11)	6.49	(0.12)	16.52	(0.21)	6.69	(0.13)
400	*3.91*	(0.07)	*	(*)	4.00	(0.15)	4.00	(0.09)	5.29	(0.10)	14.92	(0.16)	5.47	(0.09)
600	*3.05*	(0.07)	*	(*)	*	(*)	3.19	(0.08)	4.17	(0.08)	13.86	(0.13)	3.97	(0.06)
800	*2.44*	(0.05)	*	(*)	*	(*)	2.91	(0.07)	3.45	(0.07)	13.20	(0.12)	3.21	(0.06)
`satimage`: $k = 6, p = 36, N = 6435, n_{\text{test}} = 5835$														
100	*17.12*	(0.17)	20.44	(0.24)	18.48	(0.10)	18.34	(0.23)	17.29	(0.11)	19.87	(0.21)	19.05	(0.24)
200	*14.88*	(0.12)	20.22	(0.17)	17.21	(0.11)	16.54	(0.18)	15.11	(0.15)	17.57	(0.18)	16.67	(0.11)
300	*13.93*	(0.10)	20.14	(0.14)	16.67	(0.11)	16.17	(0.17)	14.05	(0.11)	16.91	(0.12)	15.60	(0.12)
400	13.27	(0.09)	20.27	(0.13)	16.15	(0.08)	15.69	(0.15)	*12.97*	(0.10)	16.09	(0.10)	14.76	(0.11)
600	12.27	(0.10)	*	(*)	15.22	(0.11)	14.97	(0.11)	*12.13*	(0.08)	15.57	(0.10)	13.97	(0.11)
800	11.91	(0.08)	*	(*)	14.47	(0.12)	14.61	(0.11)	*11.69*	(0.07)	15.47	(0.08)	13.13	(0.09)

Table 4.3: Prediction error (%) on benchmark data applications. Our multicategory kernel DWD (denoted by WZ) are compared with DWD (denoted by HLD (Huang et al., 2013)), multiclass kernel SVM (R package `SMSVM`), reinforcement angle-based multiclass SVM (R package `ramsvm`), random forest (R package `randomForest`), gradient boosting machines (R package `gbm`), and k-nearest neighbors (R package `class`). The results are averaged by 40 independent runs, and the standard error of the mean prediction error is given in parentheses. For each data, k and p are the number of classes and dimensions, and the total sample size is N. For each case, the method incurring the lowest error is marked by italics. Cases when the algorithm did not converge within ten hours are marked as "*".

n_{train}	*Prediction error (%) for the following methods:*													
	DWD WZ		*DWD HLD*		*Multiclass SVM*		*Angle-based MSVM*		*Random forest*		*Gradient boosting*		*k-nearest neighbors*	
`segmentation`: $k = 7, p = 19, N = 2310, n_{\text{test}} = 1510$														
100	18.00	(0.30)	23.09	(0.31)	17.93	(0.31)	17.34	(0.38)	*13.60*	(0.32)	16.10	(0.41)	21.89	(0.34)
200	14.04	(0.26)	21.50	(0.25)	14.04	(0.21)	14.67	(0.28)	*9.99*	(0.23)	12.15	(0.21)	15.92	(0.25)
300	12.46	(0.26)	21.30	(0.25)	11.33	(0.28)	13.21	(0.22)	*7.87*	(0.16)	11.13	(0.18)	12.70	(0.28)
400	11.15	(0.18)	21.01	(0.22)	9.72	(0.19)	12.54	(0.17)	*6.85*	(0.16)	10.57	(0.16)	11.27	(0.17)
600	9.47	(0.14)	*	(*)	7.77	(0.16)	11.85	(0.16)	*5.36*	(0.11)	9.79	(0.12)	8.82	(0.12)
800	8.44	(0.11)	*	(*)	*	(*)	11.26	(0.16)	*4.51*	(0.09)	9.38	(0.11)	7.64	(0.12)
`sensorless`: $k = 3, p = 48, N = 15957, n_{\text{test}} = 15157$														
100	0.99	(0.13)	0.16	(0.01)	0.51	(0.04)	0.60	(0.06)	*0.03*	(0.00)	0.21	(0.05)	12.73	(0.49)
200	0.31	(0.05)	0.08	(0.00)	0.31	(0.02)	0.33	(0.02)	*0.02*	(0.00)	0.09	(0.01)	6.26	(0.21)
300	0.17	(0.01)	0.06	(0.00)	0.27	(0.02)	0.33	(0.03)	*0.02*	(0.00)	0.08	(0.01)	3.88	(0.18)
400	0.14	(0.01)	0.06	(0.00)	0.25	(0.02)	0.28	(0.02)	*0.01*	(0.00)	0.08	(0.01)	2.87	(0.15)
600	0.12	(0.01)	0.06	(0.00)	0.22	(0.02)	0.20	(0.01)	*0.01*	(0.00)	0.08	(0.01)	1.84	(0.08)
800	0.10	(0.01)	0.05	(0.00)	0.19	(0.01)	0.19	(0.02)	*0.01*	(0.00)	0.07	(0.01)	1.29	(0.06)
`vowel`: $k = 11, p = 11, N = 990, n_{\text{test}} = 190$														
100	40.83	(0.69)	60.34	(0.63)	49.72	(0.68)	45.82	(0.89)	*36.46*	(0.75)	48.63	(0.72)	57.70	(0.73)
200	23.63	(0.53)	55.57	(0.61)	40.86	(0.73)	33.54	(0.64)	*21.82*	(0.61)	40.28	(0.58)	41.89	(0.61)
300	*14.93*	(0.54)	57.89	(0.12)	29.87	(0.74)	30.45	(0.63)	15.71	(0.51)	37.11	(0.60)	30.43	(0.52)
400	*9.05*	(0.38)	*	(*)	21.70	(0.68)	26.84	(0.62)	11.29	(0.48)	36.43	(0.56)	20.80	(0.51)
600	*4.03*	(0.28)	*	(*)	*	(*)	23.51	(0.44)	5.75	(0.31)	34.80	(0.50)	9.79	(0.45)
800	*2.03*	(0.17)	*	(*)	*	(*)	22.32	(0.50)	3.96	(0.27)	34.33	(0.46)	5.28	(0.25)
`waveform`: $k = 3, p = 40, N = 4999, n_{\text{test}} = 4199$														
100	*16.86*	(0.15)	18.74	(0.20)	22.55	(0.28)	22.44	(0.40)	18.74	(0.25)	18.83	(0.24)	28.37	(0.40)
200	*15.40*	(0.10)	16.46	(0.11)	19.33	(0.17)	18.75	(0.23)	16.91	(0.11)	17.06	(0.12)	25.39	(0.26)
300	*14.95*	(0.08)	15.66	(0.09)	17.60	(0.13)	17.91	(0.17)	16.29	(0.09)	16.26	(0.08)	24.28	(0.22)
400	*14.54*	(0.07)	15.12	(0.11)	16.60	(0.11)	17.35	(0.16)	15.84	(0.09)	15.97	(0.08)	23.82	(0.21)
600	*14.21*	(0.06)	14.77	(0.09)	15.62	(0.08)	16.54	(0.12)	15.42	(0.09)	15.54	(0.09)	22.69	(0.15)
800	*14.23*	(0.07)	14.72	(0.07)	15.21	(0.08)	16.24	(0.13)	15.26	(0.07)	15.56	(0.06)	22.35	(0.10)

Table 4.4: Mean computation time on benchmark data applications. We compare our multi-category kernel DWD (denoted by WZ) with DWD (denoted by HLD (Huang et al., 2013)), multiclass kernel SVM (R package `SMSVM`), reinforcement angle-based multiclass SVM (R package `ramsvm`), random forest (R package `randomForest`), gradient boosting machines (R package `gbm`), and k-nearest neighbors (R package `class`). The results are averaged by 40 independent runs. The computation time of each method includes the parameter tunes. Computations were conducted on a single-processor Intel(R) Xeon(R) central processor unit E5-2660 at 2.60 GHz. Cases when the algorithm did not converge within ten hours are marked as "*".

n_{train}	100	200	300	400	600	800	100	200	300	400	600	800
	`abalone`: $k=3, p=9, N=4177, n_{\text{test}}=3377$						`covtype`: $k=3, p=10, N=73631, n_{\text{test}}=72831$					
DWD (WZ)	3.9	15.2	42.9	100.3	462.1	967.3	0.8	3.5	7.7	13.9	63.9	154.6
DWD (HLD)	6.6	12.4	315.5	719.9	2603.9	6226.6	7.7	21.1	529.5	1111.1	3613.4	8670.1
Multiclass SVM	3.2	26.2	99.1	256.8	946.0	2342.2	5.3	23.1	79.1	204.2	771.5	1966.8
Angle-based SVM	4.3	10.2	19.8	37.9	80.2	175.6	43.1	50.6	62.8	83.4	135.2	245.8
Random forest	0.2	0.4	0.5	0.7	1.0	1.2	4.4	4.5	4.7	4.7	4.7	5.0
Gradient boosting	0.2	0.3	0.4	0.4	0.6	0.7	3.4	3.4	3.4	3.5	3.6	3.8
k-nearest neighbors	0.1	0.1	0.1	0.1	0.1	0.2	0.4	0.6	0.8	1.0	1.4	1.7
	`pendigits`: $k=10, p=16, N=10990, n_{\text{test}}=10190$						`satimage`: $k=6, p=36, N=6435, n_{\text{test}}=5835$					
DWD (WZ)	0.9	2.6	5.6	10.2	58.1	221.5	2.5	5.2	11.9	23.4	99.5	271.3
DWD (HLD)	1080.9	11139.8	*	*	*	*	311.1	2512.5	8194.5	27861.0	*	*
Multiclass SVM	98.8	1083.9	4482.8	12268.9	*	*	25.2	213.6	784.6	2009.3	7563.4	20737.1
Angle-based SVM	75.9	270.4	595.7	1054.8	2358.3	4269.0	21.3	70.1	163.5	294.6	659.6	1236.0
Random forest	0.6	0.9	1.0	1.2	1.5	1.8	0.3	0.5	0.8	1.1	1.7	2.4
Gradient boosting	1.6	2.0	2.3	2.6	3.3	3.9	0.8	1.1	1.4	1.7	2.2	2.9
k-nearest neighbors	0.1	0.1	0.2	0.2	0.3	0.5	0.1	0.1	0.2	0.3	0.4	0.7
	`segmentation`: $k=7, p=19, N=2310, n_{\text{test}}=1510$						`sensorless`: $k=3, p=48, N=15957, n_{\text{test}}=15157$					
DWD (WZ)	1.1	2.9	6.4	11.4	76.0	288.5	2.5	10.5	26.6	50.3	299.0	1085.7
DWD (HLD)	285.4	3734.8	12850.7	26370.3	*	*	6.9	17.6	632.7	1377.4	6591.7	14212.0
Multiclass SVM	43.3	401.1	1606.8	4299.3	21251.3	*	2.2	14.8	53.0	135.5	579.7	1489.1
Angle-based SVM	24.6	100.0	224.6	398.6	898.7	1672.2	9.7	15.6	24.4	38.7	79.5	164.3
Random forest	0.2	0.4	0.6	0.7	1.1	1.3	1.6	1.7	1.8	1.8	2.4	2.2
Gradient boosting	0.4	0.6	0.9	1.2	1.7	2.2	0.9	1.1	1.3	1.5	1.9	2.4
k-nearest neighbors	0.1	0.1	0.1	0.1	0.2	0.3	0.2	0.4	0.6	0.8	1.3	1.9
	`vowel`: $k=11, p=11, N=990, n_{\text{test}}=190$						`waveform`: $k=3, p=40, N=4999, n_{\text{test}}=4199$					
DWD (WZ)	1.6	4.9	10.7	19.0	61.9	143.0	2.7	12.3	32.0	60.6	335.3	959.1
DWD (HLD)	1458.4	13342.3	26280.3	*	*	*	7.8	19.3	626.7	1301.8	4039.3	9705.7
Multiclass SVM	102.7	1272.7	5416.9	17436.9	*	*	1.8	10.9	39.5	93.8	338.4	848.9
Angle-based SVM	81.6	322.1	717.6	1279.4	2879.1	5214.6	4.6	10.6	20.0	37.2	78.6	170.1
Random forest	0.2	0.4	0.5	0.7	1.0	1.4	0.3	0.6	0.9	1.2	2.1	3.0
Gradient boosting	0.4	0.6	1.0	1.3	1.9	2.5	0.4	0.6	0.7	0.9	1.3	1.7
k-nearest neighbors	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.2	0.3	0.4	0.7

Table 4.5: Rank of the prediction accuracy of each method on benchmark data applications. For each data set, the rank is given based on the average prediction accuracy of different methods on different training sizes: 100, 200, 300, 400, 600, 800. For each data, k and p are the number of classes and dimensions. For each method, the ranks of the prediction accuracy on different data sets are averaged to yield the overall rank.

	k	p	*DWD WZ*	*DWD HLD*	*Multiclass SVM*	*Angle-based MSVM*	*Random forest*	*Gradient boosting*	*k-nearest neighbors*
abalone	3	9	2	4	1	3	6	5	7
covtype	3	10	5	7	2	4	1	3	6
pendigits	10	16	2	7	3	1	4	6	5
satimage	6	36	2	7	5	4	1	6	3
segmentation	7	19	4	7	3	6	1	2	5
sensorless	3	48	5	2	4	6	1	3	7
vowel	11	11	1	7	5	4	2	6	3
waveform	3	40	1	2	5	6	3	4	7
overall			2	6	3	4	1	5	6

Table 4.6: Rank of the computation time of each method on benchmark data applications. For each data set, the rank is given based on the average prediction accuracy of different methods on different training sizes: 100, 200, 300, 400, 600, 800. For each data, k and p are the number of classes and dimensions. For each method, the ranks of the computation time on different data sets are averaged to yield the overall rank.

	k	p	*DWD WZ*	*DWD HLD*	*Multiclass SVM*	*Angle-based MSVM*	*Random forest*	*Gradient boosting*	*k-nearest neighbors*
abalone	3	9	5	7	6	4	3	2	1
covtype	3	10	4	7	6	5	3	2	1
pendigits	10	16	4	6	7	5	2	3	1
satimage	6	36	4	7	6	5	2	3	1
segmentation	7	19	4	7	6	5	2	3	1
sensorless	3	48	5	7	6	4	3	2	1
vowel	11	11	4	7	6	5	2	3	1
waveform	3	40	6	7	5	4	3	2	1
overall			4	7	6	5	2	2	1

4.6 Discussion

In this chapter we have proposed a new multicategory kernel DWD for multiclass classification. Our method is able to capture potential highly nonlinear structure of the Bayes rule and hence is more desirable than the restrictive multicategory linear DWD. We have derived an efficient accelerated PGD algorithm for fitting multicategory kernel DWD. Extensive simulations and benchmark data examples have shown that multicategory kernel DWD is a worthy competitor against the popular off-the-shelf multiclass classifiers, including the SVM, random forest, gradient boosting, and k-nearest neighbors. Therefore, we view multicategory kernel DWD as a valuable addition to the classification toolbox for real applications.

The classification problem in this chapter has equal weights on different classes. In many applications, we may face the non-standard classification problems such as imbalanced class size or unequal cost. Non-standard SVM and DWD have previously studied in Lee et al. (2004) and Qiao et al. (2010). In the future research, it will be interesting to generalize the proposal in this chapter to the non-standard case. Moreover, after establishing the Fisher consistency of multicategory kernel DWD, we would also like to establish the asymptotic convergence rates of the generalization error, which is left as a future research topic.

Chapter 5

Magic Cross-Validation Theory for Large-Margin Classification

Cross-validation (CV) is perhaps the most widely used tool for tuning supervised machine learning algorithms and estimating the generation error. CV are typically expensive to compute due to the repetition of the model fits. To lessen such computational burden, the leave-one-out lemma has been presented for the least square regression; nonetheless for classification, all existing fast CV methods are either complicated, narrowly applicable, or unable to yield exact CV estimates. In this chapter, we develop a magic formula for doing CV in the context of large-margin classification, e.g., the support vector machine (SVM) and logistic regression. We propose a magic SVM, which is a novel algorithm based on the magic CV formula to integrate the processes of fitting and tuning the SVM. As shown in extensive numerical studies, the magic SVM is significantly faster than the state-of-the-art SVM solvers.

5.1 Introduction

Cross-validation (e.g., Lachenbruch and Mickey, 1968; Stone, 1974; Geisser, 1975; Wahba and Wold, 1975; Allen, 1977; Arlot and Celisse, 2010) is most widely used in parameter selection and model assessment in regression and classification. A common variant of cross-validation (CV) is V-fold CV, which segments the original sample into V equal-

sized folds, training on every $V-1$ fold amalgamation and evaluating prediction error on the left-over fold. The average of such prediction error is CV error, which can be regarded as an estimate of the generalization error on unseen test data.

Leave-one-out cross-validation (LOOCV) is a special instance of the V-fold CV. It is referred to the case when V equals the sample size n. Under such circumstance, as much data as possible are exploited to assemble the training sets, and the remaining validation sets are mutually exclusive. LOOCV possesses appealing properties. First, Luntz and Brailovsky (1969) have shown that LOOCV yields almost unbiased estimate of the generalization error. The bias of V-fold CV decreases as V increases. Second, different data splits in V-fold CV propagate additional variability. In other words, the results given by V-fold CV with $V < n$ are random in terms of data splits, whereas the LOOCV result is deterministic due to the unique data split. The variability of V-fold CV caused by different data splits has been numerically studied by Rodríguez et al. (2010). In addition, there is a quite widespread statement saying that LOOCV has larger variance than other V-fold CVs; many researchers claimed so by incorrectly citing Efron (1983), Bailey and Elkan (1993), and Breiman (1996). However, Efron (1983) and Bailey and Elkan (1993) actually compared cross-validation with the bootstrap, and Breiman (1996) attended to a special example with unstable algorithms. In fact, much work has been done to theoretically and numerically demonstrate that the variance of LOOCV error tends to be identical to the other V-fold CVs (e.g., Burman, 1989; Kohavi, 1995; Bengio and Grandvalet, 2004; Molinaro et al., 2005; Zhang and Yang, 2015). We further substantiate this result by fitting the kernel SVM on a mixture of Gaussian models. As delineated in Figure 5.1, LOOCV has almost no bias in estimating the test error, while the bias largely increases as V decreases. We also see that the standard error of the CV error exhibits very little difference as V varies. Similar trends are also observed for kernel logistic regression (Zou and Hastie, 2005).

Due to the practical appeal of cross-validation, an algorithm should integrate both model fits and cross-validations. However, as LOOCV fits the model n times, LOOCV is computationally expensive, and the cost is especially considerable when the final model is selected along a fine grid of tuning parameters. Much work has been devoted to fast LOOCV computations. Historically, the V-fold CV was invented at the outset as a computational remedy of LOOCV (Geisser, 1975). In the least square regression, the leave-one-

out lemma (Craven and Wahba, 1979) allows for an efficient computation of LOOCV error with essentially the same cost as a single fit. The details regarding the leave-one-out lemma will be reviewed later. On the other hand, the classification does not have such universal magic formula. Many fast LOOCV methods have been proposed for specific classification problems. Among these methods, some yield the exact LOOCV error as the classic method does, for example, kernel Fisher discriminant analysis (Cawley and Talbot, 2003; Bo et al., 2006), k-nearest neighbors (Celisse and Mary-Huard, 2012), and least square support vector machines (Cawley and Talbot, 2004; Zhao and Keong, 2004; Cawley, 2006; An et al., 2007; Cawley and Talbot, 2007); while others only achieve approximated LOOCV error, e.g., support vector machines (SVM) (Wahba, 1999; Opper and Winther, 2000; Zhang and Wang, 2016), kernel logistic regression (Cawley and Talbot, 2008), and neural networks (Hansen and Larsen, 1996). Moreover, Cauwenberghs and Poggio (2001); Izbicki (2013); Joulani et al. (2015) implemented fast cross-validations for incremental training methods. Some work also theoretically approximate LOOCV error: examples include the VC-type span bounds for the SVM (Vapnik and Chapelle, 2000; Chapelle et al., 2002) and the influence functions for kernel methods (Debruyne et al., 2008; Liu et al., 2014). Unlike the leave-one-out lemma in regression, most fast LOOCV methods in classification are mathematically complicated.

In this chapter, we introduce a magic CV theory for kernel classifiers. We propose the magic SVM by applying the magic CV theory to the kernel SVM. We shall demonstrate in numerial studies that the magic SVM is significantly faster than the state-of-the-art SVM solvers. In this chapter, we specially focus on the kernel SVM. Our method can be easily generalized to other kernel classifiers, e.g., logistic regression. The magic CV theory covers LOOCV, V-fold CV, delete-v CV, among others. The recipe is surprisingly simple, but the method is extremely powerful. With much less computational cost, the magic CV exactly computes the CV error obtained by the classic CV approach, rather than an approximation.

In Section 5.2, we review the leave-one-out lemma for regression and derive the magic CV theory for kernel classifiers. We develop the magic SVM in Section 2.3. We present numerical studies are presented in Section 5.4.

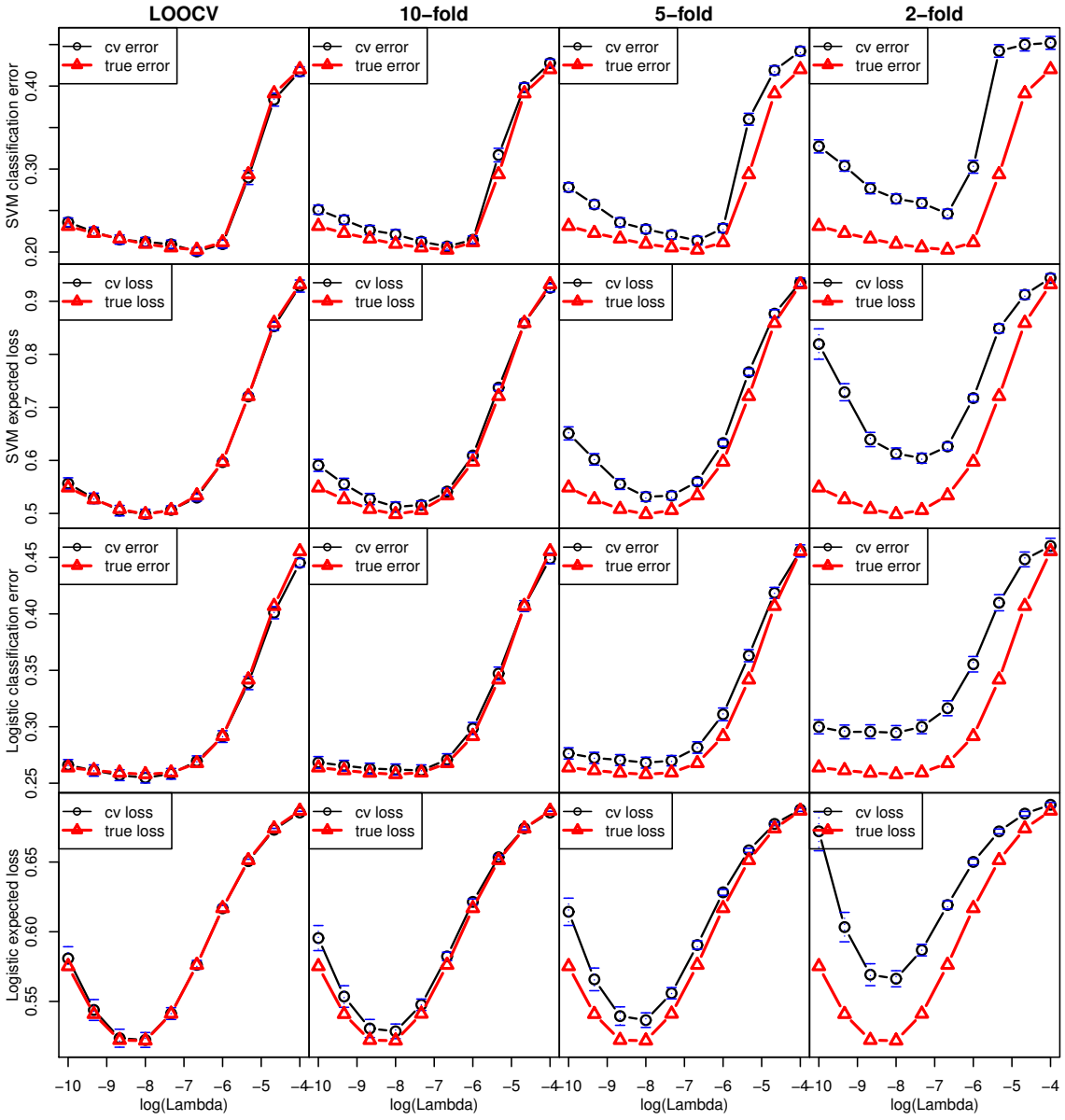

Figure 5.1: The first two rows depict the kernel SVM: the top four panels are curves of CV error (circles) and true error (triangles), and the bottom four panels are curves of CV expected loss (circles) and true expected loss (triangles). The last two rows are the same plots for kernel logistic regression. The data $(n = 100, p = 20)$ are generated from a mixture of Gaussian models (Bayes error: 13.8%.) We draw 10 centers $\boldsymbol{\mu}_k$ from $N((-1,1)^T, \boldsymbol{I})$, and we then randomly pick up one center and generate one positive-class observation from $N(\boldsymbol{\mu}_k, 4\boldsymbol{I})$. Observations from the negative class are convened similarly, with $N((1,-1)^T, \boldsymbol{I})$. The true error and the true expected loss are assessed on 10,000 observations generated independently. Errors and losses are averaged over 100 independent runs, and the standard error bars are also given.

5.2 Magic Cross-Validation

5.2.1 Review of cross-validation for regression

Consider the following model,

$$y = f(\boldsymbol{x}) + \epsilon.$$

The response $y \in \mathbb{R}$ takes continuous value and the predictor $\boldsymbol{x} \in \mathbb{R}^p$. Suppose that we are given training data $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$, and we estimate f using a generic regularization,

$$\hat{f}_\lambda = \underset{f}{\operatorname{argmin}} \left[\frac{1}{n} \sum_{i=1}^{n} (y_i - f(\boldsymbol{x}_i))^2 + \lambda P(f) \right], \tag{5.1}$$

where $P(f)$ is a penalty term, and λ is a tuning parameter varying the model complexity. Among many others, two common examples are ridge regression, when $f(\boldsymbol{x}_i) = \boldsymbol{x}_i^T \boldsymbol{\beta}$ for $\boldsymbol{\beta} \in \mathbb{R}^p$ and $P(f) = \|\boldsymbol{\beta}\|_2^2$, and smoothing splines, when $P(f) = \int f''(u)^2 du$.

Cross-validation can be used to select the tuning parameter λ in (5.1). We randomly split the training data into V disjoint sets, and a map $\tau : \{1, \ldots, n\} \to \{1, \ldots, V\}$ indicates which set the ith data point is assigned to. Typical choices of V are 5, 10, and n. When $V = n$, we customarily let $\tau(i) = i$ and the corresponding method is known as the leave-one-out cross-validation (LOOCV). Denote by

$$\hat{f}_\lambda^{[-v]} = \underset{f}{\operatorname{argmin}} \left[\frac{1}{n} \sum_{i=1, i \neq v}^{n} (y_i - f(\boldsymbol{x}_i))^2 + \lambda P(f) \right], \tag{5.2}$$

the fitted function estimated using the training data with the vth data point removed. Then the LOOCV error is

$$\mathrm{LOOCV}(\lambda) = \frac{1}{n} \sum_{i=1}^{n} \left(y_i - \hat{f}_\lambda^{[-i]}(\boldsymbol{x}_i) \right)^2,$$

and the LOOCV estimate of λ is the minimizer of $\mathrm{LOOCV}(\lambda)$.

Craven and Wahba (1979) derived the following leave-one-out lemma. The lemma

arises from the *self-stability* property, that the fitted regression model $\hat{f}$ remains unaltered when adding a new data point on the fitted surface, i.e., $(\boldsymbol{x}, \hat{f}(\boldsymbol{x}))$. Although the lemma was originally tailored for the smoothing splines, it can be readily applied on general regression problems.

Lemma 5.1
For each $v = 1, \ldots, V$, suppose $\hat{f}_\lambda^{[-v]}$ is the solution of (5.2) Let $\tilde{y}_v = \hat{f}_\lambda^{[-v]}(\boldsymbol{x}_v)$ and $\tilde{y}_i = y_i$ if $i \neq v$. Then we have

$$\hat{f}_\lambda^{[-v]} = \underset{f}{\operatorname{argmin}} \left[\frac{1}{n} \sum_{i=1}^{n} (\tilde{y}_i - f(\boldsymbol{x}_i))^2 + \lambda P(f) \right].$$

Suppose the problem (5.1) has a solution $\hat{f}_\lambda(\boldsymbol{x}_i) = \boldsymbol{H}_i^T \boldsymbol{y}$ where $\boldsymbol{H}$ is a symmetric matrix with the ith column $\boldsymbol{H}_i$ and the ith diagonal element h_{ii}. The LOOCV error has a shortcut formula:

$$\mathrm{LOOCV}(\lambda) = \frac{1}{n} \sum_{i=1}^{n} \left(y_i - \hat{f}_\lambda^{[-i]}(\boldsymbol{x}_i) \right)^2 = \frac{1}{n} \sum_{i=1}^{n} \frac{(y_i - \hat{f}_\lambda(\boldsymbol{x}_i))^2}{(1 - h_{ii})^2}.$$

□

To be self-contained, the proof of Lemma 5.1 is shown in the Appendix D.

5.2.2 Magic leave-one-out cross-validation for margin classifiers

In a binary classification problem, each y_i belongs to $\{-1, 1\}$. A margin classifier can be written in the following *loss-plus-penalty* form,

$$\hat{f}_\lambda = \underset{f}{\operatorname{argmin}} \left[\frac{1}{n} \sum_{i=1}^{n} L\left(y_i \cdot f(\boldsymbol{x}_i)\right) + \lambda P(f) \right]. \tag{5.3}$$

The classification rule for a new data point $\boldsymbol{x}_{\mathrm{new}}$ is the sign of $\hat{f}_\lambda(\boldsymbol{x}_{\mathrm{new}})$.

Popular choices of the loss functions $L(\cdot)$ are

- Logistic regression: $L(u) = \log(1 + \exp(-u))$.

- Support vector machine (SVM): $L(u) = (1-u)_+$, namely, the *hinge loss*.
- Squared SVM: $L(u) = [(1-u)_+]^2$.
- Huber SVM (Zhang, 2004; Wang et al., 2008):

$$L(u) = \begin{cases} -u, & \text{if } u \le -1, \\ (1-u)^2/4, & \text{if } -1 < u \le 1, \\ 0, & \text{if } u > 1. \end{cases}$$

- Distance weighted discrimination (DWD, Marron et al., 2007):

$$L(u) = \begin{cases} 1-u, & \text{if } u \le 1/2, \\ 1/(4u), & \text{if } u > 1/2. \end{cases}$$

An important subclass of (5.3) are *kernel classifiers*, which are associated with a reproducing kernel Hilbert space (RKHS) $\mathcal{H}_K$, being generated by a positive-definite kernel function K:

$$\hat{f}_\lambda = \underset{f \in \mathcal{H}_K}{\operatorname{argmin}} \left[\frac{1}{n} \sum_{i=1}^{n} L\left(y_i \cdot f(\boldsymbol{x}_i)\right) + \lambda \|f\|_{\mathcal{H}_K}^2 \right]. \tag{5.4}$$

A popular example of K is Gaussian radial basis function kernel, $K(\boldsymbol{x}, \boldsymbol{x}') = \exp(-\|\boldsymbol{x} - \boldsymbol{x}'\|^2/(2\sigma^2))$.

The theory of reproducing kernels (Wahba, 1990) indicates, for some $\hat{\boldsymbol{\alpha}} \in \mathbb{R}^n$, $\hat{f}(\boldsymbol{x}) = \sum_{i=1}^n \hat{\alpha}_i K(\boldsymbol{x}_i, \boldsymbol{x})$ and $\|\hat{f}\|_{\mathcal{H}_K}^2 = \sum_{i=1}^n \sum_{j=1}^n \hat{\alpha}_i K(\boldsymbol{x}_i, \boldsymbol{x}_j) \hat{\alpha}_j$. We define the positive-definite kernel matrix $\boldsymbol{K}$ with the (i,j)th element $K(\boldsymbol{x}_i, \boldsymbol{x}_j)$. Accordingly, the problem (5.4) amounts to

$$\hat{\boldsymbol{\alpha}} = \underset{\boldsymbol{\alpha} \in \mathbb{R}^n}{\operatorname{argmin}} \left[\frac{1}{n} \sum_{i=1}^{n} L\left(y_i \cdot \boldsymbol{K}_i^T \boldsymbol{\alpha}\right) + \lambda \boldsymbol{\alpha}^T \boldsymbol{K} \boldsymbol{\alpha} \right].$$

The corresponding classification rule for $\boldsymbol{x}_{\text{new}}$ is the sign of $\sum_{i=1}^n \hat{\alpha}_i K(\boldsymbol{x}_i, \boldsymbol{x}_{\text{new}})$.

Throughout this chapter, we avoid the so-called *one-class classification* (Moya and Hush, 1996; Khan and Madden, 2009), which is referred to an extreme case that the training set consists of observations from only one class. When using cross-validation, the training set under each split is also required to contain both positive and negative classes.

For each $v = 1, \ldots, n$, define the LOOCV estimator as

$$\hat{f}_\lambda^{[-v]} = \underset{f \in \mathcal{H}_K}{\operatorname{argmin}} \left[\frac{1}{n} \sum_{i=1,\ i \neq v}^{n} L\left(y_i \cdot f(\boldsymbol{x}_i)\right) + \lambda P(f) \right]. \tag{5.5}$$

We present the magic leave-one-out lemma for the margin classifiers.

Lemma 5.2

Suppose $\hat{f}_\lambda^{[-v]}$ is defined in (5.5). If we let $\tilde{y}_v = 0$ and $\tilde{y}_i = y_i$ if $i \neq v$, then we have

$$\hat{f}_\lambda^{[-v]} = \underset{f \in \mathcal{H}_K}{\operatorname{argmin}} \left[\frac{1}{n} \sum_{i=1}^{n} L\left(\tilde{y}_i \cdot f(\boldsymbol{x}_i)\right) + \lambda P(f) \right].$$

□

Proof 5.1

For any function f, we have that,

$$\begin{aligned}
\frac{1}{n} \sum_{i=1}^{n} L\left(\tilde{y}_i \cdot \hat{f}_\lambda^{[-v]}(\boldsymbol{x}_i)\right) + \lambda P(\hat{f}_\lambda^{[-v]}) =& \frac{1}{n} \sum_{i=1,\ i \neq v}^{n} L\left(y_i \cdot \hat{f}_\lambda^{[-v]}(\boldsymbol{x}_i)\right) + \frac{1}{n} L(0) + \lambda P(\hat{f}_\lambda^{[-v]}) \\
\leq& \frac{1}{n} \sum_{i=1,\ i \neq v}^{n} L\left(y_i \cdot f(\boldsymbol{x}_i)\right) + \frac{1}{n} L(0) + \lambda P(f) \\
=& \frac{1}{n} \sum_{i=1}^{n} L\left(\tilde{y}_i \cdot f(\boldsymbol{x}_i)\right) + \lambda P(f),
\end{aligned}$$

where the inequality is due to (5.5) and the two equalities arise from $\tilde{y}_v = 0$. □

Consequently, by the theory of reproducing kernels, the LOOCV (5.5) is paraphrased

as

$$\tilde{\boldsymbol{\alpha}}^{[-v]} = \underset{\boldsymbol{\alpha}\in\mathbb{R}^n}{\operatorname{argmin}} \left[\frac{1}{n} \sum_{i=1}^{n} L\left(\tilde{y}_i \cdot \boldsymbol{K}_i^T \boldsymbol{\alpha}\right) + \lambda \boldsymbol{\alpha}^T \boldsymbol{K} \boldsymbol{\alpha} \right]. \tag{5.6}$$

Lemma 5.2 can be readily generalized to the cases of V-fold CV and delete-v CV. We take V-fold CV as an example. Suppose the ith data point is allocated to the set $\tau(i)$ by randomization, then for each $v = 1, \ldots, V$, the V-fold CV estimate is defined as

$$\hat{f}_\lambda^{[-v]} = \underset{f\in\mathcal{H}_K}{\operatorname{argmin}} \left[\frac{1}{n} \sum_{\{i:\tau(i)\neq v\}} L\left(y_i \cdot f(\boldsymbol{x}_i)\right) + \lambda \|f\|_{\mathcal{H}_\mathcal{K}}^2 \right]. \tag{5.7}$$

The magic CV formula for the V-fold CV is presented as follows.

Lemma 5.3
For each $v = 1, \ldots, V$, suppose $\hat{f}_\lambda^{[-v]}$ is defined in (5.7); for a given $\tau(\cdot)$, we define $\tilde{y}_i = 0$ if $\tau(i) = v$, and $\tilde{y}_i = y_i$ if $\tau(i) \neq v$, then we have

$$\hat{f}_\lambda^{[-v]} = \underset{f\in\mathcal{H}_K}{\operatorname{argmin}} \left[\frac{1}{n} \sum_{i=1}^{n} L\left(\tilde{y}_i \cdot f(\boldsymbol{x}_i)\right) + \lambda \|f\|_{\mathcal{H}_\mathcal{K}}^2 \right]. \qquad \square$$

5.3 Magic CV Algorithms in Computing SVM

In this section, we apply the magic CV formula in solving kernel classifiers. Two most popular examples are the kernel SVM and kernel logistic regression. Since good selection of tuning parameters in kernel classifiers is of paramount importance, the algorithm should integrate computing the complete solution path and conducting cross-validation for model tunes. Owing to the repetitions in the model fits, we realize that computational burden of the state-of-the-art SVM algorithms is large. Therefore, we propose an exact smoothing principle for solving the SVM, and we then apply the magic CV formula for LOOCV. Our method is exemplified by the SVM, although it can be readily generalized to smooth losses, for instance, logistic regression.

5.3.1 Exact smoothing principle for kernel SVM

The state-of-the-art algorithm for solving SVMs is through sequential minimal optimization (SMO). The terminology SMO was coined by Platt (1999). The earliest SVM algorithms include the chunking algorithm (Boser et al., 1992) and the decomposition algorithm Osuna et al. (1997), shrinking the original problem into smaller QP sub-problems. Specifically, Osuna et al. (1997) partitioned the index set $\{1, \ldots, n\}$ into a *working set* containing free variables and another set of fixed variables, and only updated the solution on the working set in each iteration. Osuna et al. (1997) assembled the working set with arbitrary observations violating the optimality conditions, and Joachims (1999) advanced the working set selection by pursuing the steepest descent in the objective function. As an extreme case, Platt (1999) restricted the working set to only contain two elements, which are the so-called *maximal violating pair*, the two data points violating the optimality conditions most. The SMO algorithm was further refined by Keerthi et al. (2001) and Fan et al. (2005) by interpreting the maximal violation in different means.

However, the application of the magic CV formula on the SMO algorithm is infeasible, because the working set is contrived based on the fact that each y_i is 1 or -1. Since SMO requires each model fit of $\mathcal{O}(n^3)$ operations, the time complexity of the entire LOOCV procedure is $\mathcal{O}(n^4)$.

We now introduce the exact smoothing principle for solving the exact solution of the kernel SVM. We then apply the magic CV theory to our new algorithm.

Let $L(u) = (1-u)_+$ be the hinge loss in (4.15). Define $\boldsymbol{\theta} = \boldsymbol{K}\boldsymbol{\alpha}$ and then $\boldsymbol{\theta}$ and $\boldsymbol{\alpha}$ are one-to-one correspondent by the positive-definiteness of $\boldsymbol{K}$. The kernel SVM is formulated as,

$$\min_{\boldsymbol{\theta}\in\mathbb{R}^n} Q(\boldsymbol{\theta}) \equiv \min_{\boldsymbol{\theta}\in\mathbb{R}^n} \sum_{i=1}^{n} L(y_i\boldsymbol{\theta}_i) + n\lambda\boldsymbol{\theta}^T\boldsymbol{K}^{-1}\boldsymbol{\theta}. \tag{5.8}$$

To solve this problem, we first construct a decreasing sequence of δ. For each δ, we

define a smoothed hinge loss function L^δ:

$$L^\delta(u) = \begin{cases} 0 & u \geq 1+\delta, \\ \dfrac{1}{4\delta}[u-(1+\delta)]^2 & 1-\delta < u < 1+\delta, \\ 1-u & u \leq 1-\delta, \end{cases}$$

and the corresponding objective function is

$$\min_{\boldsymbol{\theta}\in\mathbb{R}^n} Q^\delta(\boldsymbol{\theta}) = \min_{\boldsymbol{\theta}\in\mathbb{R}^n} \sum_{i=1}^{n} L^\delta(y_i\boldsymbol{\theta}_i) + n\lambda\boldsymbol{\theta}^T\boldsymbol{K}^{-1}\boldsymbol{\theta}. \tag{5.9}$$

Note that we are not inventing or advocating a new loss function. The proposal of the smoothed hinge loss is only for solving the SVM. Our smoothed hinge loss is different from the Huber SVM studied by Zhang (2004); Wang et al. (2008); Yang and Zou (2013).

As $\delta \to 0$, it is trivial to see the SVM solution being approximated by the solutions of a sequence of problem (5.9). Compared to the original problem (5.8) with non-differentiable hinge loss, the problem (5.9) is much simpler to solve. However, it is well known that the non-differentiable hinge point plays an important role in delivering high prediction accuracy for the SVM. Thus, our ultimate target is to seek the exact SVM solution rather than an approximation. To this end, we present the following theorem.

Theorem 5.1

(Exact Smoothing Principle) With training data given, suppose $\boldsymbol{\theta}^{\mathrm{SVM}}$ and $\boldsymbol{\theta}^\delta$ are the unique solution of $Q(\boldsymbol{\theta})$ and $Q^\delta(\boldsymbol{\theta})$, respectively, then there exists a small $\delta^\star$ such that $\boldsymbol{\theta}^\delta = \boldsymbol{\theta}^{\mathrm{SVM}}$ when $\delta < \delta^\star$.

Theorem 5.1 declares that the exact SVM solution is virtually attained before δ reaches zero. Therefore, given a sequence of $\delta_{(d)}$ with $\delta_{(d+1)} = r\delta_{(d)}$ and $0 < r < 1$. We solve $\hat{\boldsymbol{\theta}}^{\delta_{(d)}}$ sequentially, and the entire algorithm stops when some $\hat{\boldsymbol{\theta}}^{\delta_{(d)}}$ satisfies the Karush-Kuhn-Tucker (KKT) condition of (5.8). The algorithm for solving $\hat{\boldsymbol{\theta}}^{\delta_{(d)}}$ will be introduced in the next subsection.

5.3.2 Accelerated proximal gradient algorithm

In this section, we develop an efficent algorithm for solving the smoothed SVM problem (5.9). Our algorithm is based on the proximal gradient method (Parikh and Boyd, 2014). For a given δ, the objective function in (4.15) is written as

$$Q(\boldsymbol{\theta}) = \ell^{\delta}(\boldsymbol{\theta}) + n\lambda\boldsymbol{\theta}^T\boldsymbol{K}^{-1}\boldsymbol{\theta}, \text{ where } \ell^{\delta}(\boldsymbol{\theta}) \equiv \sum_{i=1}^{n} L^{\delta}\left(y_i\theta_i\right).$$

Suppose $\boldsymbol{\theta}^{(1)}$ is an initial value. For each k, the proximal gradient method updates $\boldsymbol{\theta}^{(k+1)}$ by

$$\boldsymbol{\theta}^{(k+1)} = \underset{\boldsymbol{\theta}\in\mathbb{R}^n}{\operatorname{argmin}} \left[2n\lambda\delta\boldsymbol{\theta}^T\boldsymbol{K}^{-1}\boldsymbol{\theta} + \frac{1}{2}\left\|\boldsymbol{\theta} - \left(\boldsymbol{\theta}^{(k)} - 2\delta\nabla\ell^{\delta}(\boldsymbol{\theta}^{(k)})\right)\right\|_2^2\right]. \tag{5.10}$$

The objective function in (5.10) is quadratic in terms of $\boldsymbol{\theta}$, hence the minimizer can be solved efficiently. By setting the gradient to be zeros, we obtain from (5.10) that

$$\boldsymbol{\theta}^{(k+1)} - \boldsymbol{\theta}^{(k)} = -\left(2\lambda\boldsymbol{I} + \frac{1}{2n\delta}\boldsymbol{K}\right)^{-1}\left(\frac{1}{n}\boldsymbol{K}\nabla\ell^{\delta}(\boldsymbol{\theta}^{(k)}) + 2\lambda\boldsymbol{\theta}^{(k)}\right).$$

Define $\boldsymbol{z}^{(k)} = \nabla\ell(\boldsymbol{\theta}^{(k)})/n$. Given $\boldsymbol{\theta} = \boldsymbol{K}\boldsymbol{\alpha}$, we have the update formula for $\boldsymbol{\alpha}^{(k+1)}$:

$$\boldsymbol{\alpha}^{(k+1)} - \boldsymbol{\alpha}^{(k)} = -\boldsymbol{P}_{\lambda}^{-1}(\boldsymbol{K})\left(\boldsymbol{K}\boldsymbol{z}^{(k)} + 2\lambda\boldsymbol{K}\boldsymbol{\alpha}^{(k)}\right), \text{ where } \boldsymbol{P}_{\lambda}(\boldsymbol{K}) = 2\lambda\boldsymbol{K} + \frac{1}{2n\delta}\boldsymbol{K}\boldsymbol{K}. \tag{5.11}$$

Parikh and Boyd (2014) claimed that the proximal gradient method converges when ℓ has a Lipschitz gradient. It can be found that for the smoothed hinge loss function,

$$|\nabla\ell^{\delta}(\boldsymbol{\theta}_1) - \nabla\ell^{\delta}(\boldsymbol{\theta}_2)| \leq \frac{1}{2\delta}\|\boldsymbol{\theta}_1 - \boldsymbol{\theta}_2\|_2.$$

We can further boost the algorithm using the proximal gradient method with the Nesterov's acceleration (Nesterov, 1983, 2005, 2007; Beck and Teboulle, 2009). Consider a numerical sequence, r_k, such that $r_1 = 1$ and $r_{k+1} = (1 + \sqrt{1 + 4r_k^2})/2$. Define $\boldsymbol{\theta}^{(0)}$ and

$\boldsymbol{\theta}^{(1)}$ as initial values. For each $k = 1, 2, \ldots$, we solve $\boldsymbol{\theta}^{(k+1)}$ as

$$\boldsymbol{\theta}^{(k+1)} = \underset{\boldsymbol{\theta}\in\mathbb{R}^n}{\operatorname{argmin}} \left[n\lambda\delta\boldsymbol{\theta}^T\boldsymbol{K}^{-1}\boldsymbol{\theta} + \frac{1}{2}\left\|\boldsymbol{\theta} - \left(\bar{\boldsymbol{\theta}}^{(k)} - 2\delta\nabla\ell(\bar{\boldsymbol{\theta}}^{(k)})\right)\right\|_2^2 \right], \tag{5.12}$$

where

$$\bar{\boldsymbol{\theta}}^{(k)} = \boldsymbol{\theta}^{(k)} + \left(\frac{r_k - 1}{r_{k+1}}\right)\left(\boldsymbol{\theta}^{(k)} - \boldsymbol{\theta}^{(k-1)}\right).$$

In addition, we employ the warm-start trick in our implementation to accelerate and stabilize the algorithm (Friedman et al., 2007). Specifically, rather than fix a single value of λ, we compute the whole solution path of $\boldsymbol{\theta}$ over a decreasing sequence $\{\lambda_l\}$. For each l, we adopt the solution $\hat{\boldsymbol{\theta}}_{[l]}$ as the initial value to solve $\hat{\boldsymbol{\theta}}_{[l+1]}$. With the warm-start, empirical results indicate that b, how many iterates of (5.11), is usually below 100. For a given value λ, the matrix inversion requires $\mathcal{O}(n^3)$ operations and each update (5.11) is $\mathcal{O}(n^2)$, so the integrated algorithm is $\mathcal{O}(n^3 + bn^2)$.

5.3.3 Magic LOOCV for kernel SVM

When LOOCV is applied on the kernel SVM, the classic mechanism requires fitting models n times with each observation alternatively removed. For each $v = 1, \ldots, n$, denote by $\boldsymbol{K}^{[-v]}$ the kernel matrix with the vth row and column removed, and the $\boldsymbol{K}_i^{[-v]}$ notation indicates the ith column of $\boldsymbol{K}^{[-v]}$. The vth leave-one-out estimate is

$$\hat{\boldsymbol{\alpha}}^{[-v]} = \underset{\boldsymbol{\alpha}\in\mathbb{R}^{n-1}}{\operatorname{argmin}} \left[\frac{1}{n}\sum_{i=1,\ i\neq v}^{n} L\left(y_i \cdot \boldsymbol{\alpha}^T\boldsymbol{K}_i^{[-v]}\right) + \lambda\boldsymbol{\alpha}^T\boldsymbol{K}^{[-v]}\boldsymbol{\alpha} \right].$$

As $\boldsymbol{K}^{[-v]}$ differs for each v, $\boldsymbol{P}_\lambda(\boldsymbol{K}^{[-v]})$ needs to be inverted individually, so each fold has approximately $\mathcal{O}(n^3 + bn^2)$ operations. The complexity of the LOOCV for each fixed λ is $\mathcal{O}(n^4 + bn^3)$.

To save the computational cost, we implant the magic CV formula. By Lemma 5.2, the leave-one-out estimate $\boldsymbol{\alpha}^{[-v]}$ can be obtained as (5.6). We notice the accelerated proximal gradient algorithm introduced in the last section works for any value of $\boldsymbol{y}$, instead of only ± 1. Therefore, for each fold, we construct the response $\tilde{\boldsymbol{y}}$, and use the algorithm to solve (5.6).

In Algorithm 4, we summarize the magic LOOCV in the kernel SVM. Since all folds share the same kernel matrix $\boldsymbol{K}$, the matrix $\boldsymbol{P}_\lambda(\boldsymbol{K})$ is inverted only once. Hence, the time complexity of Algorithm 4 is $\mathcal{O}(bn^3)$, being far less than the time complexity of the classic LOOCV.

5.3.4 Magic LOOCV for kernel logistic regression

Another important margin classifier is logistic regression. We now solve kernel logistic regression using the fixed-Hessian Newton-Raphson method. The objective function is

$$Q(\boldsymbol{\alpha}) = \frac{1}{n}\sum_{i=1}^{n} L(y_i \cdot \boldsymbol{K}_i^T\boldsymbol{\alpha}) + \lambda\boldsymbol{\alpha}\boldsymbol{K}\boldsymbol{\alpha}, \text{ where } L(u) = \log\left(1+\exp(-u)\right).$$

With an initial value $\boldsymbol{\alpha}^{(1)}$, the Newton-Raphson update with respect to $\boldsymbol{\alpha}$ is

$$\begin{aligned}\boldsymbol{\alpha}^{(k+1)} &= \boldsymbol{\alpha}^{(k)} - \left(\nabla^2 Q(\boldsymbol{\alpha}^{(k)})\right)^{-1}\nabla Q(\boldsymbol{\alpha}^{(k)}) \\ &= \boldsymbol{\alpha}^{(k)} - \left(2\lambda\boldsymbol{K} + \frac{1}{n}\boldsymbol{K}\boldsymbol{W}\boldsymbol{K}\right)^{-1}(\boldsymbol{K}\boldsymbol{z}^{(k)} + 2\lambda\boldsymbol{K}\boldsymbol{\alpha}^{(k)}),\end{aligned}$$

where $z_i^{(k)} = y_i \cdot L'(y_i \cdot \boldsymbol{K}_i^T\boldsymbol{\alpha})/n$ and $\boldsymbol{W}$ is a diagonal matrix with the ith element of $L''(y_i \cdot \boldsymbol{K}_i^T\boldsymbol{\alpha})$. We observe that each $L''(y_i \cdot \boldsymbol{K}_i^T\boldsymbol{\alpha}) \le 1/4$, hence the update with the fixed Hessian becomes

$$\boldsymbol{\alpha}^{(k+1)} = \boldsymbol{\alpha}^{(k)} - \left(2\lambda\boldsymbol{K} + \frac{1}{4n}\boldsymbol{K}\boldsymbol{K}\right)^{-1}(\boldsymbol{K}\boldsymbol{z}^{(k)} + 2\lambda\boldsymbol{K}\boldsymbol{\alpha}^{(k)}).$$

We can also employ the Nesterov's acceleration for the fixed-Hessian Newton-Raphson method. Like (5.12), with $\boldsymbol{\alpha}^{(k-1)}$ and $\boldsymbol{\alpha}^{(k)}$ being computed, we have that

$$\boldsymbol{\alpha}^{(k+1)} = \bar{\boldsymbol{\alpha}}^{(k)} - \left(2\lambda\boldsymbol{K} + \frac{1}{4n}\boldsymbol{K}\boldsymbol{K}\right)^{-1}\left(\boldsymbol{K}\bar{\boldsymbol{z}}^{(k)} + 2\lambda\boldsymbol{K}\bar{\boldsymbol{\alpha}}^{(k)}\right),$$

in which

$$\bar{\boldsymbol{\alpha}}^{(k)} = \boldsymbol{\alpha}^{(k)} + \left(\frac{r_k - 1}{r_{k+1}}\right)(\boldsymbol{\alpha}^{(k)} - \boldsymbol{\alpha}^{(k-1)}), \text{ and } \bar{\boldsymbol{z}}^{(k)} = \frac{1}{n}y_i \cdot L(y_i \cdot \boldsymbol{K}_i^T\bar{\boldsymbol{\alpha}}^{(k)}).$$

Algorithm 4 Magic leave-one-out cross-validation for kernel SVM

Require: $\boldsymbol{y}$, $\boldsymbol{K}$, λ, and r.

for $v = 1, \ldots, n$ **do**
  Let $\tilde{y}_i = y_i$ if $i \neq v$, and $\tilde{y}_v = 0$.
  Initialize $\tilde{\boldsymbol{\alpha}}^{[-v]}$.
  Initialize δ. Define L^{δ}.
  repeat
    Update $\tilde{\boldsymbol{\alpha}}'^{[-v]} \leftarrow \tilde{\boldsymbol{\alpha}}^{[-v]}$.
    Compute $\boldsymbol{P}_{\lambda}^{-1}(\boldsymbol{K}) = (2\lambda \boldsymbol{K} + \frac{1}{2n\delta}\boldsymbol{K}\boldsymbol{K})^{-1}$.
    repeat
      Compute $\tilde{\boldsymbol{z}} = (\tilde{z}_1, \ldots, \tilde{z}_n)^T$, with $\tilde{z}_i = \tilde{y}_i L'^{\delta}(y_i \cdot \boldsymbol{K}_i \tilde{\boldsymbol{\alpha}}^{[-v]})/n$.
      Update $\tilde{\boldsymbol{\alpha}}^{[-v]} \leftarrow \tilde{\boldsymbol{\alpha}}^{[-v]} - \boldsymbol{P}_{\lambda}^{-1}(\boldsymbol{K})\left(\boldsymbol{K}\tilde{\boldsymbol{z}} + 2\lambda \boldsymbol{K}\tilde{\boldsymbol{\alpha}}^{[-v]}\right)$.
    until the convergence condition is met.
    Update $\delta \leftarrow r\delta$
  until the KKT condition check of SVM is passed.
end for

The magic CV used for logistic regression is similar to the one used for the kernel SVM. We summarize the magic LOOCV in Algorithm 5.

Algorithm 5 Magic leave-one-out cross-validation for kernel logistic regression

Require: $\boldsymbol{y}$, $\boldsymbol{K}$, and λ.

Compute $\boldsymbol{P}_{\lambda}^{-1}(\boldsymbol{K}) = (2\lambda \boldsymbol{K} + \frac{1}{4n}\boldsymbol{K}\boldsymbol{K})^{-1}$.
for $v = 1, \ldots, n$ **do**
  Let $\tilde{y}_i = y_i$ if $i \neq v$, and $\tilde{y}_v = 0$.
  Initialize $\tilde{\boldsymbol{\alpha}}^{[-v]}$.
  repeat
    Compute $\tilde{\boldsymbol{z}} = (\tilde{z}_1, \ldots, \tilde{z}_n)^T$, with $\tilde{z}_i = \tilde{y}_i L'(\tilde{y}_i \cdot \boldsymbol{K}_i^T \tilde{\boldsymbol{\alpha}}^{[-v]})/n$.
    Update $\tilde{\boldsymbol{\alpha}}^{[-v]} \leftarrow \tilde{\boldsymbol{\alpha}}^{[-v]} - \boldsymbol{P}_{\lambda}^{-1}(\boldsymbol{K})\left(\boldsymbol{K}\tilde{\boldsymbol{z}} + 2\lambda \boldsymbol{K}\tilde{\boldsymbol{\alpha}}^{[-v]}\right)$.
  until the convergence condition is met.
end for

5.4 Numerical Studies

In this section, we illustrate the algorithmic performance and statistical properties of the magic cross-validation with simulated data and benchmark data sets. We implement our

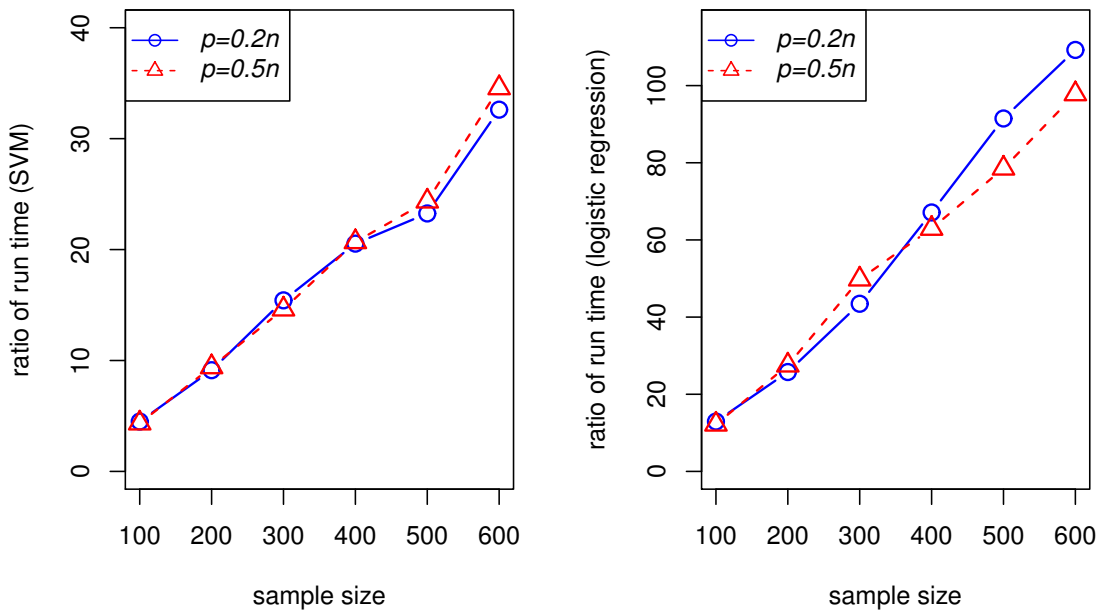

Figure 5.2: Ratios of the run time without using the magic CV formula to the run time with using the magic CV formula. The left panel is for the kernel SVM and the right one is for kernel logistic regression. All the time includes selecting the tuning parameter from 100 candidate values. Each time is averaged over 20 independent runs. Computations were conducted on an Intel Core i5 M560 (2.67 GHz) processor.

algorithms in an R package `magicsvm`. Our package handles logistic regression as well.

5.4.1 Performance of Magic Cross-Validation

We implement the magic SVM with the exact smoothing algorithm as well as the magic CV formula. In this section, we demonstrate that the magic CV formula has significant effect in reducing the computation time, and we also show the magic SVM has large advantages over `kernlab` (Karatzoglou et al., 2004) and `e1071` (Meyer et al., 2015), which are the two most popular R packages for solving the SVM.

We generated data from a mixture Gaussian model. Define $\boldsymbol{\mu}_+ = (1, \ldots, 1, 0, \ldots, 0)$ and $\boldsymbol{\mu}_- = (0, \ldots, 0, 1, \ldots, 1)$, half of which are ones and the other half are zeros. In each

example, the positive class arose from a mixture Gaussian distribution $\sum_{k=1}^{10} 0.1N(\boldsymbol{\mu}_{k+}, 4\boldsymbol{I})$ with each $\boldsymbol{\mu}_{k+}$ drawn from $N(\boldsymbol{\mu}_{+}, \boldsymbol{I})$, and likewise the negative class was assembled by $\sum_{k=1}^{10} 0.1N(\boldsymbol{\mu}_{k-}, 4\boldsymbol{I})$ with each $\boldsymbol{\mu}_{k-}$ from $N(\boldsymbol{\mu}_{-}, \boldsymbol{I})$. We included twelve examples, with the sample sizes $n = \{100, 200, \ldots, 600\}$ and the dimensions $p = 0.2n$ and $p = 0.5n$. In each example, we fitted the kernel SVM and kernel logistic regression with Gaussian kernels, and conducted LOOCV for model tunes.

Figure 5.2 depicts ratios of the run time without and with the magic CV formula. For both the SVM and logistic regression, we observe the ratio linearly increases against the sample size. When $n = 600$, the magic CV formula boosts the logistic regression above 100 times faster. We also verified that the results obtained with and without the magic CV formula are almost identical, with slight differences caused by machine errors and implementations.

We next compare `magicsvm` to `kernlab` and `e1071`. With the same simulation settings, we fitted kernel SVMs with Gaussian kernels using the three packages. Table 5.1 records the total run time. With λ selected by `magicsvm`, we refit the kernel SVM using `kernlab` and `e1071` and computed the objective values, $Q(\boldsymbol{\theta})$ in (5.8).

In Table 5.1, we observe the objective values of three packages are the same, with only one exception in the case when $n = 200$ and $p = 100$. Furthermore, we see that the exact smoothing principle equipped with the magic CV formula is always the fastest. For example, when $n = 400$ and $p = 200$, the package `e1071` spent more than two and a half hours to complete the entire computation; as a comparison, within six minutes, the same results were obtained by `magicsvm`. In conclusion, our method yields almost the same results as `kernlab` and `e1071`, but our method is significantly faster.

5.4.2 Benchmark Data Applications

In this section, we examine the performance of `magicsvm` on eight benchmark data sets obtained from the UCI machine learning repository (Dua, D. and Karra Taniskidou, E., 2017). We randomly split each data into a training set and a test set with equal sizes. On the training data, using our package `magicsvm` and the packages `kernlab` and `e1071`, we fitted the kernel SVM and conducted both LOOCV and 10-fold CV for selecting the

Table 5.1: Time comparison of three R solvers of kernel SVM: `magicsvm`, `kernlab`, and `e1071`. The run time includes the leave-one-out cross-validations to select tuning parameter. Using the tuning parameter selected by `magicsvm`, the corresponding objective values obtained by three R solvers are exhibited and the respective standard errors are given in parentheses. Computations were conducted on an Intel Core i5 M560 (2.67 GHz) processor.

n	p	time (seconds)			objective value (5.8)		
		magicsvm	kernlab	e1071	magicsvm	kernlab	e1071
200	40	45.059	319.026	617.379	0.640	0.640	0.640
					(0.011)	(0.011)	(0.011)
	100	39.461	504.113	900.292	0.599	0.599	0.675
					(0.014)	(0.014)	(0.073)
300	60	122.958	967.405	2305.041	0.631	0.631	0.631
					(0.010)	(0.010)	(0.010)
	150	107.102	1680.196	3443.444	0.611	0.611	0.611
					(0.018)	(0.017)	(0.017)
400	80	373.881	2279.869	6232.798	0.629	0.629	0.629
					(0.012)	(0.012)	(0.012)
	200	343.342	4506.013	9473.093	0.608	0.608	0.608
					(0.015)	(0.015)	(0.015)

tuning parameter. In Table 5.2, we recorded total computation time including the model fits and tunes, and we assessed the prediction error on the test sets. As a comparison, we fitted kernel logistic regression using our package `magicsvm`. We observe that in both LOOCV and 10-fold CV, the magic SVM has significant advantages over `kernlab` and `e1071` in computation time, while the three packages deliver similar prediction accuracy. We also find that the prediction accuracy of the kernel SVM is overall better than kernel logistic regression. When LOOCV is used, in all examples except for the LSVT data, the magic SVM has better accuracy than the logistic regression: when 10-fold CV is used, the magic SVM is always better the logistic regression.

Table 5.2: Run time and prediction error on eight UCI benchmark data. The run time includes fitting kernel SVM (magicSVM, kernlab, e1071) and kernel logistic regression (magicLogistic) and tuning using LOOCV and 10-fold CV. All the time and error are averaged over 20 independent runs, with standard errors given in parentheses. Computations were carried out on an Intel Core i5 M560 (2.67 GHz) processor.

data	method	leave-one-out CV			10-fold CV		
		time (sec)	error (%)		time (sec)	error (%)	
arrhythmia	magicSVM	46.199	24.093	(0.602)	5.971	24.425	(0.559)
$n = 452$	kernlab	755.316	24.447	(0.602)	34.586	24.646	(0.648)
$p = 191$	e1071	1544.779	24.137	(0.602)	60.231	24.447	(0.655)
	magicLogistic	5.458	24.779	(0.665)	0.998	24.867	(0.570)
australian	magicSVM	63.798	14.058	(0.372)	5.903	14.014	(0.386)
$n = 690$	kernlab	440.839	14.058	(0.388)	13.649	14.188	(0.396)
$p = 14$	e1071	887.315	14.203	(0.371)	22.910	14.058	(0.347)
	magicLogistic	14.575	14.942	(0.337)	2.694	14.623	(0.261)
hepatitis	magicSVM	1.664	14.464	(1.044)	0.365	14.286	(1.012)
$n = 112$	kernlab	22.837	14.643	(0.966)	4.638	14.732	(0.838)
$p = 18$	e1071	20.781	14.732	(0.933)	3.646	15.268	(0.864)
	magicLogistic	0.106	15.893	(1.011)	0.041	15.714	(1.033)
LSVT	magicSVM	3.185	15.873	(0.620)	0.314	16.587	(0.810)
$n = 126$	kernlab	111.767	15.635	(0.766)	16.934	16.587	(0.742)
$p = 309$	e1071	232.434	16.270	(0.763)	37.136	16.508	(0.872)
	magicLogistic	0.265	15.317	(0.703)	0.054	18.492	(0.965)
musk	magicSVM	97.780	10.798	(0.415)	8.381	10.777	(0.419)
$n = 476$	kernlab	708.587	11.113	(0.433)	32.418	11.092	(0.373)
$p = 166$	e1071	2078.586	10.777	(0.420)	73.621	10.735	(0.405)
	magicLogistic	6.197	12.794	(0.478)	1.153	11.996	(0.378)
SAfrica	magicSVM	53.558	28.939	(0.616)	4.145	29.654	(0.859)
$n = 462$	kernlab	342.820	29.589	(0.750)	15.582	29.502	(0.753)
$p = 65$	e1071	791.532	29.307	(0.692)	26.657	29.286	(0.785)
	magicLogistic	5.202	29.827	(0.776)	0.940	30.281	(0.765)
sonar	magicSVM	14.151	17.885	(0.749)	1.362	18.125	(0.781)
$n = 208$	kernlab	80.996	18.798	(0.973)	8.301	17.837	(0.756)
$p = 60$	e1071	134.681	18.942	(1.351)	11.413	18.125	(0.897)
	magicLogistic	0.844	20.529	(1.180)	0.158	20.769	(0.945)
valley	magicSVM	129.080	2.096	(0.186)	8.375	1.980	(0.168)
$n = 606$	kernlab	899.347	2.195	(0.232)	30.701	1.947	(0.193)
$p = 100$	e1071	2314.652	2.096	(0.209)	72.831	2.046	(0.214)
	magicLogistic	11.586	3.284	(0.273)	2.001	3.366	(0.269)

References

Aizerman, A., Braverman, E., and Rozoner, L. (1964), "Theoretical foundations of the potential function method in pattern recognition learning," *Automation and remote control*, 25, 821–837.

Alizadeh, F. and Goldfarb, D. (2004), "Second-order cone programming," *Mathematical Programming, Series B*, 95(1), 3–51.

Allen, D. (1977), "The relationship between variable selection and data augmentation and a method of prediction," *Technometrics*, 16(1), 125–127.

Allwein, E. L., Schapire, R. E., and Singer, Y. (2000), "Reducing multiclass to binary: a unifying approach for margin classifiers," *Journal of Machine Learning Research*, 1, 113–141.

An, S., Liu, W., and Venkatesh, S. (2007), "Fast cross-validation algorithms for least squares support vector machine and kernel ridge regression," *Pattern Recognition*, 40(8), 2154–2162.

Anthony, M. and Bartlett, P. (1999), *Neural Network Learning: Theoretical Foundations.*, Cambridge University Press.

Arlot, S. and Celisse, A. (2010), "A survey of cross-validation procedures for model selection," *Statistics surveys*, 4, 40–79.

Bailey, T. and Elkan, C. (1993), "Estimating the accuracy of learned concepts," *Proceedings of International Joint Conference on Artificial Intelligence*, 896–900.

Barsoum, E., Zhang, C., Ferrer, C. C., and Zhang, Z.(2016), "Training deep networks for facial expression recognition with crowd-sourced label distribution," *Proceedings of the 18th ACM International Conference on Multimodal Interaction*, 279–283.

Bartlett, P., Jordan, M., and McAuliffe, J. (2006), "Convexity, classification, and risk bounds," *Journal of the American Statistical Association*, 101(473), 138–156.

Bartlett, P. and Shawe-Taylor, J. (1999), "Generalization performance of support vector machines and other pattern classifiers", *Advances in Kernel Methods–Support Vector Learning*, 43–54.

Beck A. and Teboulle, M. (2009), "A fast iterative shrinkage-thresholding algorithm for linear inverse problems," *SIAM Journal on Imaging Sciences*, 2(1), 183–202.

Bengio, Y. and Grandvalet, Y. (2004), "No unbiased estimator of the variance of K-fold cross-validation," *Journal of Machine Learning Research*, 5, 1089–1105.

Bo, L., Wang, L., and Jiao, L. (2006), "Feature scaling for kernel fisher discriminant analysis using leave-one-out cross validation," *Neural Computation*, 18(4), 961–978.

Bojarski, M., Yeres, P., Choromanska, A., Choromanski, K., Firner, B., Jackel, L., and Muller, U. (2017), "Explaining how a deep neural network trained with end-to-end learning steers a car." Preprint. (Available from https://arxiv.org/pdf/1704.07911.)

Boser, B., Guyon, I, and Vapnik, V. (2004), "A training algorithm," *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*, 144–152.

Boyd, S. and Vandenberghe, L. (2004), *Convex Optimization*, Cambridge University Press.

Bradley, P. and Mangasarian, O. (1998), "Feature selection via concave minimization and support vector machines," in *Machine Learning Proceedings of the Fifteenth International Conference (ICML'98)*, Citeseer, 82–90.

Bredensteiner, E. J. and Bennett, K. P. (1999), "Multicategory classification by support vector machines," *Computational Optimization and Applications*, 12, 53–79.

Breiman, L. (1996), "Heuristics of instability and stabilization in model selection," *Annals of Statistics*, 24(6), 2350–2383.

Breiman, L. (2001), "Random forests," *Machine Learning*, 45(1), 5–32.

Burman, P. (1989), "A comparative study of ordinary cross-validation, v-fold cross-validation and the repeated learning-testing methods," *Biometrika*, 76(3), 503–514.

Cauwenberghs, G. and Poggio, T. (2001), "Incremental and decremental support vector machine learning," *Advances in Neural Information Processing Systems*, 13, 409-415.

Cawley, G. (2006), "Leave-one-out cross-validation based model selection criteria for weighted LS-SVMs," *International Joint Conference on Neural Networks*, IEEE, 1661–1668.

Cawley, G. and Talbot, N.L. (2003), "Efficient leave-one-out cross-validation of kernel fisher discriminant classifiers," *Pattern Recognition*, 36(11), 2585–2592.

Cawley, G. and Talbot, N.L. (2004), "Fast exact leave-one-out cross-validation of sparse least-squares support vector machines," *Neural networks*, 17(10), 1467–1475.

Cawley, G. and Talbot, N.L. (2007), "Preventing over-fitting during model selection via Bayesian regularisation of the hyper-parameters," *Journal of Machine Learning Research*, 8, 841–861.

Cawley, G. and Talbot, N.L. (2008), "Efficient approximate leave-one-out cross-validation for kernel logistic regression," *Machine Learning*, 71(2-3), 243–264.

Celisse, A. and Mary-Huardm T. (2012), "Exact Cross-Validation for kNN: application to passive and active learning in classification," *Journal de la Société Française de Statistique*, 152(3), 83–97.

Chapelle, O., Vapnik, V., Bousquet, O., and Mukherjee, S. (2002), "Choosing multiple parameters for support vector machines," *Machine learning*, 46, 131–159.

Chen, T., Li, M., Li, Y., Lin, M., Wang, N., Wang, M., Xiao, T., Xu, B., Zhang, C., Zhang, Z. (2015), "Mxnet: a flexible and efficient machine learning library for heterogeneous distributed systems," arXiv: 1512.01274.

Chen, J., Wang, Y., Yoho, S.E., Wang, D., and Healy, E.W. (2016), "Large-scale training to increase speech intelligibility for hearing-impaired listeners in novel noises," *Journal of the Acoustical Society of America*, 139(5), 2604–2612.

Chen, C., Sun, D.F., and Toh, K.C. (2017), "An efficient inexact symmetric gauss-seidel based majorized ADMM for high-dimensional convex composite conic programming," *Mathematical Programming*, 161(1), 237–270.

Crammer, K. and Singer, Y. (2001), "On the algorithmic implementation of multiclass kernel-based vector machines," *Journal of Machine Learning Research*, 2, 265–292.

Craven, P. and Wahba, G. (1979), "Smoothing noisy data with spline functions," *Numerische Mathematik*, 31(4), 377–403.

Debruyne, M., Hubert, M., and Suykens, J.A. (2008), "Model selection in kernel based regression using the influence function," *Journal of Machine Learning Research*, 9, 2377–2400.

De Leeuw, J. and Heiser, W. (1977), "Convergence of correction matrix algorithms for multidimensional scaling", 735–752.

Dietterich, T. G. and Bakiri, G. (1995), "Solving multiclass learning problems via error-correcting output codes," *Journal of Artificial Intelligence Research*, 2, 263–286.

Dong, L. and Shan, J. (2013), "A comprehensive review of earthquake-induced building damage detection with remote sensing techniques," *ISPRS Journal of Photogrammetry and Remote Sensing*, 84, 85–99.

Donoho, D. L. and Johnston, I. M. (1994), "Ideal spatial adaptation by wavelet shrinkage," *Biometrika*, 81(3), 425–455.

Dua, D. and Karra Taniskidou, E. (2017), "UCI Machine Learning Repository," School of Information and Computer Science, University of California at Irvine, Irvine. (Available from http://archive.ics.uci.edu/ml.)

Efron, B. (1983), "Estimating the error rate of a prediction rule: improvement on cross-validation," *Journal of American Statistical Association*, 78(382), 316–331.

Fan, R., Chen, P., and Lin, C. (2005), "Working set selection using second order information for training support vector machines," *Journal of Machine Learning Research*, 6, 1889–1918.

Fan, J. and Fan, Y. (2008), "High dimensional classification using featured annealed independence rules," *Annals of Statistics*, 36(6), 2605–2637.

Fernández-Delgado, M., Cernadas, E., Barro, S., and Amorim, D. (2014), "Do we need hundreds of classifiers to solve real world classification problems?" *Journal of Machine Learning Research*, 15, 3133–3181.

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

Friedman, J., Hastie, T., Höfling, H., and Tibshirani, R. (2007), "Pathwise coordinate optimization," *Annals of Applied Statistics*, 1(2), 302–332.

Fu, S., Zhang, S., and Liu, Y. (2018), "Adaptively weighted large-margin angle-based classifiers," *Journal of Multivariate Analysis*, 166, 282–299.

Geisser, S. (1975), "The predictive sample reuse method with applications," *Journal of American Statistical Association*, 70(350), 320–328.

Golub, G., Heath, M., and Wahba, G. (1979), "Generalized cross-validation as a method for choosing a good ridge parameter," *Technometrics*, 21(2), 215–223.

Graham, K., de las Morenas, A., Tripathi, A., King, C., Kavanah, M., Mendez, J., Stone, M., Slama, J., Miller, M., Antoine, G., Willers, H., Sebastiani, P., and Rosenberg, C. (2010), "Gene expression in histologically normal epithelium from breast cancer patients and from cancer-free lemmahylactic mastectomy patients shares a similar profile," *British Journal of Cancer*, 102(8), 1284–1293.

Guermeur, Y. (2002), "Combining discriminant models with new multi-class SVMs," *Pattern Analysis & Applications*, 5(2), 168–179.

Hall, P., Marron, J.S., and Neeman, A. (2005), "Geometric representation of high dimensions, low sample size data," *Journal of the Royal Statistical Society, Series B*, 67(3), 427–444.

Hansen, J.H. and Hasan, T. (2015), "Speaker recognition by machines and humans: A tutorial review," *IEEE Signal Processing Magazine*, 32(6), 74–99.

Hansen, L. and Larsen, J. (1996), "Linear unlearning for cross-validation," *Advances in Computational Mathematics*, 5(1), 269–280.

Haralick, R.M. and Shanmugam, K. (1973), "Textural features for image classification," *IEEE Transactions on Systems, Man, and Cybernetics*, 6, 610–621.

Hastie, T. and Tibshirani, R. (1998), "Classification by pairwise coupling," *Annals of Statistics*, 26(2), 451–471.

Hastie, T., Tibshirani, R., and Friedman, J. (2009), *The Elements of Statistical Learning: Prediction, Inference, and Data Mining*, 2nd edition, Springer-Verlag.

Hsieh, C., Si, S., Dhillon. I. (2014), "A divide-and-conquer solver for kernel support vector machines," *International Conferences on Machine Learning*, 2014, 566–574.

Hsu, C. and Lin, C. (2002), "A comparison of methods for multiclass support vector machines," *IEEE Transactions on Neural Networks*, 13(2), 415–425.

Huang, H., Lu, X., Liu, Y., Haaland, P., and Marron, J.S. (2012), "R/DWD: distance-weighted discrimination for classification, visualization and batch adjustment," *Bioinformatics*, 28(8), 1182–1183.

Huang, H., Liu, Y., Du. Y., Perou, C., Hayes, D., Todd, M., and Marron, J.S. (2013), "Multiclass distance-weighted discrimination," *Journal of Computational and Graphical Statistics*, 22(4), 953–969.

Hunter, D. and Lange, K. (2004), "A tutorial on MM algorithms," *American Statistician*, 58(1), 30–37.

Hunter, D. and Li, R. (2005), "Variable selection using MM algorithms," *Annals of Statistics*, 33(4), 1617–1642.

Izbicki, M. (2013), "Algebraic classifiers: a generic approach to fast cross-validation, online training, and parallel training," *Proceedings of the 30th International Conference on Machine Learning*, 648–656.

Jaakkola, T. and Haussler, D. (1999), "Probabilistic kernel regression models," *Proceedings of the 1999 Conference on AI and Statistics*, 126, 00–04.

James, G. and Hastie, T. (1998), "The error coding method and PICTs," *Journal of Computational and Graphical Statistics*, 7(3), 377–387.

Joachims, T. (1999), "Making large-scale SVM learning practical," *Advances in Kernel Methods-Support Vector Learning*, 11, 41–56.

Joulani, P., Györgym, A., and Szepesvári, C. (2015), "Fast Cross-Validation for Incremental Learning," *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence*, 3597–3604.

Karatzoglou, A., Smola, A., Hornik, K., and Zeileis, A.(2004), "kernlab – An S4 Package for Kernel Methods in R," *Journal of Statistical Software*, 11(9), 1–20.

Keerthi, S., Shevade, S., Bhattacharyya, C., and Murthy, K. (2001), "Improvements to Platt's SMO algorithm for SVM classifier design," *Neural Computation*, 13(3), 637–649.

Khan, S. and Madden, G. (2009), "A survey of recent trends in one class classification," *Artificial Intelligence and Cognitive Science*, Springer Berlin Heidelberg, 188–197.

Kohavi, R. (1995), "A study of cross-validation and bootstrap for accuracy estimation and model selection," *International Joint Conference on Artificial Intelligence*, 14(2), 1137–1145.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012), "Imagenet classification with deep convolutional neural networks," *Advances in Neural Information Processing Systems*, 1097–1105.

Lachenbruch, P. and Mickey, M. (1968), "Estimation of error rates in discriminant analysis," *Technometrics*, 10(1), 1–11.

Lam, X., Marron, J.S., Sun, D., and Toh, K.C. (2017), "Fast algorithms for large scale generalized distance weighted discrimination," arXiv: 1604.05473.

Lange, K., Hunter, D., and Yang, I. (2000), "Optimization transfer using surrogate objective functions," *Journal of Computational and Graphical Statistics*, 9(1), 1–20.

Lange, K. and Zhou, H. (2014), "MM algorithms for geometric and signomial programming," *Mathematical Programming*, 143(1-2), 339–356.

Lee, Y., Lin, Y., Wahba, G. (2004), "Multicategory support vector machines: Theory and application to the classification of microarray data and satellite radiance data," *Journal of the American Statistical Association*, 99(465), 67–81.

Li, X.D., Sun, D.F., and Toh, K.C. (2016), "A Schur complement based semi-proximal ADMM for convex quadratic conic programming and extensions," *Mathematical Programming*, 155(1), 333–373.

Liaw, A. and Wiener, M. (2002), "Classification and regression by randomForest," *R News*, 2(3), 18–22.

Lillesand, T., Kiefer, R., and Chipman, J. (2014), *Remote sensing and image interpretation*, 2nd edition, John Wiley & Sons.

Lin, Y., Lee, Y., and Wahba, G. (2002), "Support vector machines for classification in nonstandard situations,"*Machine Learning*, 46, 191–202.

Lin, Y. (2002), "Support vector machines and the Bayes rule in classification," *Data Mining and Knowledge Discovery*, 6(3), 259–275.

Lin, Y. (2004), "A note on margin-based loss functions in classification," *Statistics & Probability Letters*, 68(1), 73–82.

Liu, Y. (2007), "Fisher consistency of multicategory support vector machines," *Proceedings of the Eleventh International Conference on Artificial Intelligence and Statistics*, 291–298.

Liu, Y. and Yuan, M. (2011), "Reinforced multicategory support vector machines,"*Journal of Computational and Graphical Statistics*, 20(4), 901–919.

Liu, Y. and Shen, X. (2006), "Multicategory ψ-Learning,"*Journal of the American Statistical Association*, 101(474), 500–509.

Liu, Y., Zhang, H., and Wu, Y. (2011), "Hard or soft classification? Large-margin unified machines," *Journal of the American Statistical Association*, 106(493), 166–177.

Liu, P., Han, S., Meng, Z., and Tong, Y. (2012), "Facial expression recognition via a boosted deep belief network," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 1805–1812.

Liu, L., Liu, Y., and Zhu, H. (2018), "SMAC: Spatial multi-category angle-based classifier for high-dimensional neuroimaging data," *NeuroImage*, 175, 230–245.

Liu, Y., Jiang, S., and Liao, S. (2014), "Efficient approximation of cross-validation for kernel methods using Bouligand influence function," *Proceedings of the 31st International Conference on Machine Learning* , 324–332.

Luntz, A. and Brailovsky, V. (1969), "On estimation of characters obtained in statistical procedure of recognition (in Russian)," *Technicheskaya Kibernetica*, 3(6), 6–12.

Marron, J.S., Todd, M., and Ahn, J. (2007), "Distance weighted discrimination," *Journal of the American Statistical Association*, 102(480), 1267–1271.

Marron, J.S. (2013), "Smoothing, functional data analysis, and distance weighted discrimination software," http://www.unc.edu/~marron/marron_software.html.

Marron, J.S. (2015), "Distance-weighted discrimination," *Wiley Interdisciplinary Reviews: Computational Statistics*, 7(2), 109–114.

Maulik, U. and Chakraborty, D. (2017), "Remote sensing image classification: a survey of support-vector-machine-based advanced techniques," *IEEE Geoscience and Remote Sensing Magazine*, 5(1), 33–52.

Meyer, D., Dimitriadou, E., Hornik, K, Weingessel, A., and Leisch, F. (2015), "Misc Functions of the Department of Statistics, Probability Theory Group."

Micchelli, C., Xu, Y., and Zhang, H. (2006), "Universal kernels," *Journal of Machine Learning Research*, 7, 2651–2667.

Molinaro, A., Simon, R., and Pfeiffer, R. (2005), "Prediction error estimation: a comparison of resampling methods," *Bioinformatics*, 21(15), 3301–3307.

Moya, M. and Hush, D. (1996), "Network constraints and multi-objective optimization for one-class classification," *Neural Networks*, 9(3), 463–474.

Nesterov, Y. (1983), "A method of solving a convex programming problem with convergence rate $O(1/k^2)$," *Soviet Mathematics Doklady*, 27(2), 372–376.

Nesterov, Y. (2005), "Smooth minimization of non-smooth functions," *Mathematical programming*, 103(1), 127–152.

Nesterov, Y. (2007), "Gradient methods for minimizing composite objective function," *No. CORE Discussion Papers*, 76, UCL.

Nesterov, Y. (2013), "Gradient methods for minimizing composite functions," *Mathematical Programming*, 140(1), 125–161.

Opper, M. and Winther, O. (2000), "Gaussian processes for classification: Mean-field algorithms," *Neural Computation*, 12(11), 2655–2684.

Osuna, E., Freund, R., and Girosi, F. (1997), "An improved training algorithm for support vector machines," *Neural Networks for Signal Processing VII, Proceedings of the 1997 IEEE Workshop*, 276–285.

Parikh, N. and Boyd, S. (2014), "Proximal algorithms," *Foundations and Trends in Optimization*, 1(3), 123–231.

Platt, J. (1999), "Fast training of support vector machines using sequential minimal optimization," *Advances in Kernel Methods-Support Vector Learning*, 12, 185–208.

Qiao, X., Zhang, H., Liu, Y., Todd, M., Marron, J.S. (2010), "Weighted distance weighted discrimination and its asymptotic properties," *Journal of the American Statistical Association*, 105(489), 401–414.

Qiao, X. and Zhang, L. (2015a), "Distance-weighted support vector machine," *Statistics and Its Interface*, 8(3), 331–345.

Qiao, X. and Zhang, L. (2015b), "Flexible high-dimensional classification machines and their asymptotic properties," *Journal of Machine Learning Research*, 16, 1547–1572.

Rabiner, L.(1989), "A tutorial on hidden Markov models and selected applications in speech recognition," *Proceedings of the IEEE*, 77(2), 257–286.

Rabiner, L. and Juang, B. (1993), "Fundamentals of speech recognition," 14, Englewood Cliffs: PTR Prentice Hall.

Ridgeway, G. (2017), "gbm: Generalized boosted regression models," R package 2.9.0.

Rodríguez, J., Pérez, A., and Lozano, J. (2010), "Sensitivity analysis of k-fold cross validation in prediction error estimation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(3), 569–575.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A.C., and Fei-Fei, L. (2015), "Imagenet large scale visual recognition challenge," *International Journal of Computer Vision*, 115(3), 211–252.

Shawe-Taylor, J. and Cristianini, N. (2000), "Margin distribution and soft margin", *Advances in Kernel Methods–Support Vector Learning*, 349–358.

Singh, D., Febbo, P., Ross, K., Jackson, D., Manola, J., Ladd, C., Tamayo, P., Renshaw, A., D'Amico, A., Richie, J. (2002), "Gene expression correlates of clinical prostate cancer behavior," *Cancer cell*, 1(2), 203–209.

Steinwart, I. (2001), "On the influence of the kernel on the consistency of support vector machines," *Journal of Machine Learning Research*, 2, 67–93.

Steinwart, I., Hush, D., and Scovel, C. (2006), "An explicit description of the reproducing kernel Hilbert spaces of Gaussian RBF kernels," *IEEE Transactions on Information Theory*, 52(10), 4635–4643.

Stone, M. (1974), "Cross-validatory choice and assessment of statistical predictions," *Journal of the royal statistical society. Series B*, 36(2), 111–147.

Sun, H., Craig, B., and Zhang, L. (2017), "Angle-based multicategory distance-weighted SVM," *Journal of Machine Learning Research*, 18(85), 1–21.

Takashima, Y., Takiguchi, T., Ariki, Y., and Omori, K. (2017), "Audio-visual speech recognition for a person with severe hearing loss using deep canonical correlation analysis," *Proc. of the 1st Conference on Challenges in Hearing Assistive Technology*, Stockholm, Sweden.

Tibshirani, R., Bien, J., Friedman, J. Hastie, T., Simon, N., Taylor, J., and Tibshirani, R. J. (2010), "Strong rules for discarding predictors in lasso-type problems," *Journal of the Royal Statistical Society, Series B*, 74(2), 245–266.

Tsanas, A., Little, M., Fox, C., and Ramig, L. (2014), "Objective automatic assessment of rehabilitative speech treatment in Parkinson's disease," *IEEE Transactions on Neural Systmes and Rehabilitation Engineering*, 22(1), 181–190.

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

Vapnik, V. (1998), *Statisitcal Learning Theory*, Wiley.

Vapnik, V. and Chapelle, O. (2000), "Bounds on error expectation for support vector machines," *Neural Computation*, 12(9), 2013–2036.

Venables, W. and Ripley, B. (2002), *Modern Applied Statistics with S*, 4th edition, Springer.

Wahba, G. (1990), *Spline Models for Observational Data*, 59, SIAM.

Wahba, G. and Wold, S. (1975), "A completely automatic French curve: Fitting spline functions by cross validation," *Communications in Statistics-Theory and Methods*, 4(1), 1–17.

Wahba, G., Gu, C., Wang, Y., and Campbell, R. (1994), "Soft classification, aka risk estimation, via penalized log likelihood and smoothing spline analysis of variance," In *Santa fe Institute Studies in the Sciences of Complexity-Proceeding Vol*, 20, 331–331.

Wahba, G. (1999), "Support vector machines, reproducing kernel Hilbert spaces and the randomized GACV," *Advances in Kernel Methods-Support Vector Learning*, 6, 69–87.

Wang, D. (2017), "Deep learning reinvents the hearing aid," *IEEE Spectrum*, 54(3), 32–37.

Wang, L. Zhu, J., and Zou, H.(2006), "The doubly regularized support vector machine," *Statistica Sinica*, 16(2), 589–616.

Wang, L., Zhu, J., and Zou, H. (2008), "Hybrid huberized support vector machines for microarray classification and gene selection," *Bioinformatics*, 24(3), 412–419.

Weston, J. and Watkins, C. (1999), "Support vector machines for multi-class pattern recognition," *Proceedings of the Seventh European Symposium on Artificial Neural Networks*, 219–224.

Wu, T. and Lange, K. (2010), "The MM Alternative to EM," *Statistical Science*, 25(4), 492–505.

Xu, H., Gao, Y., Yu, F., and Darrel, T. (2016), " End-to-end Learning of Driving Models from Large-scale Video Datasets." Preprint. (Available from https://arxiv.org/abs/1612.01079.)

Yang, Y. and Zou, H. (2013), "An efficient algorithm for computing the HHSVM and its generalizations," *Journal of Computational and Graphical Statistics*, 22(2), 396–415.

Yu, D. and Deng, L. (2016), *Automatic Speech Recognition*, London: Springer.

Zhang, T. (2004), "Statistical behavior and consistency of classification methods based on convex risk minimization," *Annals of Statistics*, 32(1), 56–134.

Zhang, H. H., Ahn, J., Lin, X., and Park, C. (2006), "Gene selection using support vector machines with nonconvex penalty," *Bioinformatics*, 22(1), 88–95.

Zhang, L. and Lin, X. (2013), "Some considerations of classification for high dimension low-sample size data," *Statistical Methods in Medical Research*, 22, 536–549.

Zhang, C. and Liu, Y. (2014), "Multicategory angle-based large-margin classification," *Biometrika*, 101(3), 625–640.

Zhang, Y., Duchi, J., and Wainwright, M. "Divide and conquer kernel ridge regression: a distributed algorithm with minimax optimal rates," *Journal of Machine Learning Research*, 16, 3299–3340.

Zhang, C., Liu, Y., Wang, J., and Shen, X. (2016), "Reinforced angel-based multicategory support vector machines," *Journal of Computational and Graphical Statistics*, 25(3), 806–825.

Zhang, J. and Wang, S. (2016), "A fast leave-one-out cross-validation for SVM-like family," *Neural Computing and Applications* , 27(6), 1717–1730.

Zhang, Y. and Yang, Y. (2015), "Cross-validation for selecting a model selection procedure," *Journal of Econometrics*, 187, 95–112.

Zhao, Y. and Keong, K.C. (2004), "Fast leave-one-out evaluation and improvement on inference for LS-SVMs," *Proceedings of the 17th International Conference on Pattern Recognition*, (3) 494–497.

Zhang, C., Pam, M., Fu, S., and Liu, Y. (2017), "Robust multicategory support vector machines using difference convex algorithm," *Mathematical Programming*, 1–29.

Zhou, H. and Lange, K. (2010), "MM algorithms for some discrete multivariate distributions," *Journal of Computational and Graphical Statistics*, 19(3), 645–665.

Zhu, J., Rosset, S., Hastie, T., and Tibshirani, R. (2004), "1-norm support vector machines," *The Annual Conference on Neural Information Processing Systems*, 16(1), 49–56.

Zou, H. and Hastie, T. (2005), "Regularization and variable selection via the elastic-net," *Journal of the Royal Statistical Society, Series B*, 67(2), 301–320.

Zou, H. (2006), "The adaptive lasso and its oracle properties," *Journal of American Statistical Association*, 101(476), 1418–1429.

Zou, H. and Li, R. (2008), "One-step sparse estimates in nonconcave penalized likelihood models," *Annals of Statistics*, 36(4), 1509-1533.

Zou, H. and Zhang, H. H. (2009), "On the adaptive elastic-net with a diverging number of parameters," *Annals of Statistics*, 37(4), 1733–1751.

Zou, H., Zhu, J., Hastie, T. (2008), "New Multicategory boosting algorithms based on multicategory Fisher-consistent losses," *Annals of Applied Statistics*, 2(4), 1290–1306.

Appendix A

Proof of Chapter 2

Proof of Lemma 2.1

Write $v_i = y_i(\omega_0 + \boldsymbol{x}_i^T\boldsymbol{\omega})$ and $G(\eta_i) = 1/(v_i+\eta_i)^q + c\eta_i$. The objective function of (2.4) can be written as $\sum_{i=1}^n G(\eta_i)$. We next minimize (2.4) over η_i for every fixed i by computing the first-order and the second-order derivatives of $G(\eta_i)$:

$$G'(\eta_i) = -\frac{q}{(v_i+\eta_i)^{q+1}} + c = 0 \Rightarrow v_i + \eta_i = \left(\frac{q}{c}\right)^{\frac{1}{q+1}},$$
$$G''(\eta_i) = \frac{q(q+1)}{(v_i+\eta_i)^{q+2}} > 0.$$

If $v_i > (\frac{q}{c})^{\frac{1}{q+1}}$, then $G'(\eta_i) > 0$ for all $\eta_i \geq 0$, and $\eta_i^\star = 0$ is the minimizer. If $v_i \leq (\frac{q}{c})^{\frac{1}{q+1}}$, then $\eta_i^\star = (\frac{q}{c})^{\frac{1}{q+1}} - v_i$ is the minimizer as $G'(\eta^\star) = 0$ and $G''(\eta^\star) > 0$.

By plugging in the minimizer $\eta_i^\star$ into $\sum_{i=1}^n G(\eta_i)$, we obtain

$$\min_{\omega_0,\boldsymbol{\omega}} \sum_{i=1}^n \tilde{V}_q\left(y_i(\omega_0 + \boldsymbol{x}_i^T\boldsymbol{\omega})\right), \text{ subject to } \boldsymbol{\omega}^T\boldsymbol{\omega} = 1; \tag{A.1}$$

$$\tilde{V}_q(v) = \begin{cases} \left(\frac{q}{c}\right)^{-\frac{q}{q+1}} + c\left(\frac{q}{c}\right)^{\frac{1}{q+1}} - cv, & \text{if } v \leq \left(\frac{q}{c}\right)^{\frac{1}{q+1}}, \\ \frac{1}{v^q}, & \text{if } v > \left(\frac{q}{c}\right)^{\frac{1}{q+1}}. \end{cases}$$

We now simplify (A.1). Suppose $t = (\frac{q}{q+1})(\frac{q}{c})^{-\frac{1}{q+1}}$ and $t_1 = (\frac{1}{q+1})(\frac{q}{c})^{\frac{q}{q+1}}$. We define

$V_q(u) = t_1 \cdot \tilde{V}_q(u/t)$ for each q,

$$V_q(u) = \begin{cases} 1-u, & \text{if } u \le \dfrac{q}{q+1}, \\ \dfrac{1}{u^q}\dfrac{q^q}{(q+1)^{q+1}}, & \text{if } u > \dfrac{q}{q+1}. \end{cases}$$

By setting $\beta_0 = t \cdot \omega_0$ and $\boldsymbol{\beta} = t \cdot \boldsymbol{\omega}$, we find that (A.1) becomes

$$\min_{\beta_0, \boldsymbol{\beta}} \sum_{i=1}^{n} V_q\left(y_i(\beta_0 + \boldsymbol{x}_i^T\boldsymbol{\beta})\right), \text{ subject to } \boldsymbol{\beta}^T\boldsymbol{\beta} = t^2,$$

which can be further transformed to (2.5) with λ and t one-to-one correspondent.

Proof of Lemma 2.2

We first prove (2.7). We observe that $0 < V_q''(u) = \frac{1}{u^{q+2}}\frac{q^{q+1}}{(q+1)^q} < \frac{(q+1)^2}{q}$, for any $u > \frac{q}{q+1}$. Also $V_q'(u)$ is continuous on $[\frac{q}{q+1}, \infty)$ and differentiable on $(\frac{q}{q+1}, \infty)$.

If both u_1 and $u_2 > \frac{q}{q+1}$, then the mean value theorem implies that there exists $u^{\star\star} > \frac{q}{q+1}$:

$$\frac{|V_q'(u_1) - V_q'(u_2)|}{|u_1 - u_2|} = |V_q''(u^{\star\star})| < \frac{(q+1)^2}{q}. \tag{A.2}$$

If $u_1 > \frac{q}{q+1}$ and $u_2 \le \frac{q}{q+1}$, then $V_q'(u_2) = V_q'\left(\frac{q}{q+1}\right) = -1$. The mean value theorem implies that there exists $u^{\star\star} > \frac{q}{q+1}$ satisfying

$$\frac{|V_q'(u_1) - V_q'(u_2)|}{|u_1 - u_2|} \le \frac{|V_q'(u_1) - V_q'(\frac{q}{q+1})|}{|u_1 - \frac{q}{q+1}|} = |V_q''(u^{\star\star})| < \frac{(q+1)^2}{q}. \tag{A.3}$$

If both u_1 and $u_2 \le \frac{q}{q+1}$, $V_q'(u_1) = V_q'(u_2) = -1$. It is trivial that

$$\frac{|V_q'(u_1) - V_q'(u_2)|}{|u_1 - u_2|} = 0 < \frac{(q+1)^2}{q}. \tag{A.4}$$

By (A.2), (A.3), and (A.4), we prove (2.7).

We now prove (2.8). Let $\nu(a) \equiv \dfrac{(q+1)^2}{2q}a^2 - V_q(a)$. From (2.7), it is not hard to show

that $\nu'(a) = \dfrac{(q+1)^2}{q}a - V_q'(a)$ is strictly increasing. Therefore $\nu(a)$ is a strictly convex function, and its first-order condition, $\nu(t) > \nu(\tilde{t}) + \nu'(\tilde{t})(t - \tilde{t})$, verifies (2.8) directly.

Proof of Lemma 2.3

Given that $\eta(\boldsymbol{x}) = P(Y = 1|\boldsymbol{X} = \boldsymbol{x})$, we have that $E_{\boldsymbol{X}Y}\left[V_q(Yf(\boldsymbol{X}))\right] \equiv E_{\boldsymbol{X}}\zeta(f(\boldsymbol{X}))$:

$$\begin{aligned}\zeta(f(\boldsymbol{x})) &\equiv \eta(\boldsymbol{x})V_q(f(\boldsymbol{x})) + [1-\eta(\boldsymbol{x})]V_q(-f(\boldsymbol{x})) \\ &= \begin{cases} \eta(\boldsymbol{x})\dfrac{1}{f(\boldsymbol{x})^q}\dfrac{q^q}{(q+1)^{q+1}} + [1-\eta(\boldsymbol{x})][1+f(\boldsymbol{x})], & \text{if } f(\boldsymbol{x}) > \dfrac{q}{q+1}, \\ \eta(\boldsymbol{x})[1-f(\boldsymbol{x})] + [1-\eta(\boldsymbol{x})][1+f(\boldsymbol{x})], & \text{if } -\dfrac{q}{q+1} \le f(\boldsymbol{x}) \le \dfrac{q}{q+1}, \\ \eta(\boldsymbol{x})[1-f(\boldsymbol{x})] + [1-\eta(\boldsymbol{x})]\dfrac{1}{[-f(\boldsymbol{x})]^q}\dfrac{q^q}{(q+1)^{q+1}}, & \text{if } f(\boldsymbol{x}) < -\dfrac{q}{q+1}. \end{cases}\end{aligned}$$

For each given $\boldsymbol{x}$, we take both $f(\boldsymbol{x})$ and $\eta(\boldsymbol{x})$ as scalars and hereby write them as f and η respectively. We then take $\zeta(f) = \zeta(f(\boldsymbol{x}))$ as a function of f and compute the derivative with respect to f:

$$\frac{\partial\zeta(f)}{\partial f} = \begin{cases} -\eta\dfrac{1}{f^{q+1}}\dfrac{q^{q+1}}{(q+1)^{q+1}} + 1 - \eta, & \text{if } f > \dfrac{q}{q+1}, \\ 1 - 2\eta, & \text{if } -\dfrac{q}{q+1} \le f \le \dfrac{q}{q+1}, \\ -\eta + (1-\eta)\dfrac{1}{(-f)^{q+1}}\dfrac{q^{q+1}}{(q+1)^{q+1}}, & \text{if } f < -\dfrac{q}{q+1}. \end{cases}$$

We see that (1) when $\eta > 0.5$, $\partial\zeta(f)/\partial f = 0$ only when $f = \tilde{f} \equiv \frac{q}{q+1}\left(\frac{\eta}{1-\eta}\right)^{\frac{1}{q+1}}$, and (2) when $\eta < 0.5$, $\partial\zeta(f)/\partial f = 0$ only when $f = \tilde{f} \equiv -\frac{q}{q+1}\left(\frac{1-\eta}{\eta}\right)^{\frac{1}{q+1}}$. Hence the convexity of the function ζ implies that $\tilde{f}$ is the minimizer of $\zeta(f)$.

Proof of Lemma 2.4

As $\tilde{f}(\boldsymbol{x})$ was defined in (2.14), we see that for each $\boldsymbol{x}$,

$$\begin{aligned}\zeta\left(\tilde{f}(\boldsymbol{x})\right) &\equiv \eta(\boldsymbol{x})V_q\left(\tilde{f}(\boldsymbol{x})\right) + [1-\eta(\boldsymbol{x})]V_q\left(-\tilde{f}(\boldsymbol{x})\right) \\ &= \begin{cases} \eta(\boldsymbol{x}) + [1-\eta(\boldsymbol{x})]^{\frac{1}{q+1}}\eta(\boldsymbol{x})^{\frac{q}{q+1}}, & \text{if } \eta(\boldsymbol{x}) \le 1/2, \\ 1-\eta(\boldsymbol{x}) + \eta(\boldsymbol{x})^{\frac{1}{q+1}}[1-\eta(\boldsymbol{x})]^{\frac{q}{q+1}}, & \text{if } \eta(\boldsymbol{x}) > 1/2, \end{cases} \\ &= \frac{1}{2}\Big(1-|2\eta(\boldsymbol{x})-1|\Big) + \frac{1}{2}\Big(1+|2\eta(\boldsymbol{x})-1|\Big)^{\frac{1}{q+1}}\Big(1-|2\eta(\boldsymbol{x})-1|\Big)^{\frac{q}{q+1}}.\end{aligned}$$

For $a \in [0,1]$, we define $\gamma(a)$ and compute its first-order derivative as follows,

$$\begin{aligned}\gamma(a) &\equiv 1 - \frac{1}{2}(1-a) - \frac{1}{2}(1+a)^{\frac{1}{q+1}}(1-a)^{\frac{q}{q+1}} - \frac{q}{q+1}a, \\ \gamma'(a) &= \frac{1}{2} - \frac{1}{2(q+1)}\left(\frac{1-a}{1+a}\right)^{\frac{q}{q+1}} + \frac{q}{2(q+1)}\left(\frac{1+a}{1-a}\right)^{\frac{1}{q+1}} - \frac{q}{q+1} \\ &= \left[\frac{1}{2(q+1)} - \frac{1}{2(q+1)}\left(\frac{1-a}{1+a}\right)^{\frac{q}{q+1}}\right] + \left[\frac{q}{2(q+1)} + \frac{q}{2(q+1)}\left(\frac{1+a}{1-a}\right)^{\frac{1}{q+1}} - \frac{q}{q+1}\right] \ge 0.\end{aligned}$$

Hence for each $a \in [0,1]$, $\gamma(a) \ge \gamma(0) = 0$. For each $\boldsymbol{x}$, let $a = |2\eta(\boldsymbol{x})-1|$ and we see that

$$1 - \zeta\left(\tilde{f}(\boldsymbol{x})\right) \ge \frac{q}{q+1}|2\eta(\boldsymbol{x})-1|.$$

By $R(f) = E_{\boldsymbol{X}Y}[Y \ne \operatorname{sign}(f(\boldsymbol{X})] = E_{\{\boldsymbol{X}:f(\boldsymbol{X})\ge 0\}}[1-\eta(\boldsymbol{X})] + E_{\{\boldsymbol{X}:f(\boldsymbol{X})\le 0\}}\eta(\boldsymbol{X})$, we obtain

$$\begin{aligned}R(\hat{f}_n) - R(f^\star) &= E_{\{\boldsymbol{X}:\hat{f}_n(\boldsymbol{X})\ge 0,\ f^\star(\boldsymbol{X})<0\}}[1-2\eta(\boldsymbol{X})] + E_{\{\boldsymbol{X}:\hat{f}_n(\boldsymbol{X})\le 0,\ f^\star(\boldsymbol{X})>0\}}[2\eta(\boldsymbol{X})-1] \\ &\le E_{\{\boldsymbol{X}:\hat{f}_n(\boldsymbol{X})f^\star(\boldsymbol{X})\le 0\}}|2\eta(\boldsymbol{X})-1| \\ &\le \frac{q+1}{q}E_{\{\boldsymbol{X}:\hat{f}_n(\boldsymbol{X})f^\star(\boldsymbol{X})\le 0\}}\left[1-\zeta\left(\tilde{f}(\boldsymbol{X})\right)\right].\end{aligned} \tag{A.5}$$

Since $f^\star(\boldsymbol{X})$ and $\tilde{f}(\boldsymbol{X})$ share the same sign, $\hat{f}_n(\boldsymbol{X})f^\star(\boldsymbol{X}) \le 0$ implies that $\hat{f}_n(\boldsymbol{X})\tilde{f}(\boldsymbol{X}) \le 0$. When $\hat{f}_n(\boldsymbol{X})\tilde{f}(\boldsymbol{X}) \le 0$, 0 is between $\hat{f}_n(\boldsymbol{X})$ and $\tilde{f}(\boldsymbol{X})$, and thus the convexity of ζ

indicates that $\zeta(\tilde{f}(\boldsymbol{X})) \leq \zeta(0) = 1 \leq \zeta(\hat{f}_n(\boldsymbol{X}))$. From (A.5), we conclude that

$$\begin{aligned} R(\hat{f}_n) - R(f^\star) &\leq \frac{q+1}{q} E_{\{\boldsymbol{X}:\hat{f}_n(\boldsymbol{X})f^\star(\boldsymbol{X})\leq 0\}} \left[\zeta\left(\hat{f}_n(\boldsymbol{X})\right) - \zeta\left(\tilde{f}(\boldsymbol{X})\right)\right] \\ &\leq \frac{q+1}{q} E_{\boldsymbol{X}} \left[\zeta\left(\hat{f}_n(\boldsymbol{X})\right) - \zeta\left(\tilde{f}(\boldsymbol{X})\right)\right] \\ &= \frac{q+1}{q} E_{\boldsymbol{X}Y} \left[V_q\left(Y\hat{f}_n(\boldsymbol{X})\right) - V_q\left(Y\tilde{f}(\boldsymbol{X})\right)\right] \\ &= \frac{q+1}{q}(\varepsilon_A + \varepsilon_E). \end{aligned}$$

Proof of Theorem 2.1

Part (1). We first show that when $\mathcal{H}_K$ is induced by a universal kernel, the approximation error $\varepsilon_A = 0$. By definition, we need to show for any $\epsilon > 0$, there exists $f_\epsilon \in \mathcal{H}_K$ such that

$$\left| E_{\boldsymbol{X}Y} V_q\left(Y f_\epsilon(\boldsymbol{X})\right) - E_{\boldsymbol{X}Y} V_q\left(Y\tilde{f}(\boldsymbol{X})\right) \right| < \epsilon. \tag{A.6}$$

We first use truncation to consider a truncated version of $\tilde{f}$. For any $\delta \in (0, 0.5)$, we define

$$f_\delta(\boldsymbol{X}) = \begin{cases} \frac{q}{q+1}\left(\frac{1-\delta}{\delta}\right)^{\frac{1}{q+1}}, & \text{if } \eta(\boldsymbol{X}) > 1-\delta, \\ \tilde{f}(\boldsymbol{X}), & \text{if } \delta \leq \eta(\boldsymbol{X}) \leq 1-\delta, \\ -\frac{q}{q+1}\left(\frac{\delta}{1-\delta}\right)^{\frac{1}{q+1}}, & \text{if } \eta(\boldsymbol{X}) < \delta. \end{cases}$$

We have that

$$0 \leq E_{\boldsymbol{X}Y} V_q\left(Y f_\delta(\boldsymbol{X})\right) - E_{\boldsymbol{X}Y} V_q\left(Y\tilde{f}(\boldsymbol{X})\right) = \kappa_+ + \kappa_-,$$

where

$$\begin{aligned}\kappa_+ =& E_{\boldsymbol{X}:\eta(\boldsymbol{X})>1-\delta}\left[\eta(\boldsymbol{X})V_q(f_\delta(\boldsymbol{X})) + (1-\eta(\boldsymbol{X}))V_q(-f_\delta(\boldsymbol{X}))\right] \\ & - E_{\boldsymbol{X}:\eta(\boldsymbol{X})>1-\delta}\left[\eta(\boldsymbol{X})V_q\left(\tilde{f}(\boldsymbol{X})\right) + (1-\eta(\boldsymbol{X}))V_q\left(-\tilde{f}(\boldsymbol{X})\right)\right], \\ \kappa_- =& E_{\boldsymbol{X}:\eta(\boldsymbol{X})<\delta}\left[\eta(\boldsymbol{X})V_q(f_\delta(\boldsymbol{X})) + (1-\eta(\boldsymbol{X}))V_q(-f_\delta(\boldsymbol{X}))\right] \\ & - E_{\boldsymbol{X}:\eta(\boldsymbol{X})<\delta}\left[\eta(\boldsymbol{X})V_q\left(\tilde{f}(\boldsymbol{X})\right) + (1-\eta(\boldsymbol{X}))V_q\left(-\tilde{f}(\boldsymbol{X})\right)\right].\end{aligned}$$

Since $V_q(f_\delta(\boldsymbol{X})) < V_q(-f_\delta(\boldsymbol{X}))$ when $\eta(\boldsymbol{X}) > 1-\delta$,

$$\begin{aligned}\kappa_+ <& E_{\boldsymbol{X}:\eta(\boldsymbol{X})>1-\delta}\left[(1-\delta)V_q(f_\delta(\boldsymbol{X})) + \delta V_q(-f_\delta(\boldsymbol{X}))\right] \\ & - E_{\boldsymbol{X}:\eta(\boldsymbol{X})>1-\delta}\left[\eta(\boldsymbol{X})V_q\left(\tilde{f}(\boldsymbol{X})\right) + (1-\eta(\boldsymbol{X}))V_q\left(-\tilde{f}(\boldsymbol{X})\right)\right] \\ =& \left[\delta + (1-\delta)^{\frac{1}{q+1}}\delta^{\frac{q}{q+1}}\right] - E_{\boldsymbol{X}:\eta(\boldsymbol{X})>1-\delta}\left[1-\eta(\boldsymbol{X}) + \eta(\boldsymbol{X})^{\frac{1}{q+1}}(1-\eta(\boldsymbol{X}))^{\frac{q}{q+1}}\right].\end{aligned}$$

We notice that $(1-a) + a^{\frac{1}{q+1}}(1-a)^{\frac{q}{q+1}}$ is a continuous function in terms of $a \in (0,1)$. Since $\eta(\boldsymbol{X}) > 1-\delta$ implies that $|\eta(\boldsymbol{X}) - (1-\delta)| < \delta$, we conclude that for any given $\epsilon > 0$, there exists a sufficiently small δ such that $\kappa_+ < \epsilon/6$. We can also obtain $\kappa_- < \epsilon/6$ in the same spirit. Therefore,

$$0 \le E_{\boldsymbol{X}Y}V_q\left(Yf_\delta(\boldsymbol{X})\right) - E_{\boldsymbol{X}Y}V_q\left(Y\tilde{f}(\boldsymbol{X})\right) \le \kappa_+ + \kappa_- < \epsilon/3. \tag{A.7}$$

By Lusin's Theorem, there exists a continuous function $\varrho(\boldsymbol{X})$ such that $P(\varrho(\boldsymbol{X}) \neq f_\delta(\boldsymbol{X})) \le \epsilon(q+1)/(6q)$. Notice that $\sup_{\boldsymbol{X}}|f_\delta(\boldsymbol{X})| \le q/(q+1)$. Define

$$\tau(\boldsymbol{X}) = \begin{cases} \varrho(\boldsymbol{X}), & \text{if } |\varrho(\boldsymbol{X})| \le \dfrac{q}{q+1}, \\ \dfrac{q}{q+1}\cdot\dfrac{\varrho(\boldsymbol{X})}{|\varrho(\boldsymbol{X})|}, & \text{if } |\varrho(\boldsymbol{X})| > \dfrac{q}{q+1}, \end{cases}$$

then $P(\tau(\boldsymbol{X}) \neq f_\delta(\boldsymbol{X})) \le \epsilon(q+1)/(6q)$ as well. Hence

$$\begin{aligned}\left|E_{\boldsymbol{X}Y}V_q\left(Yf_\delta(\boldsymbol{X})\right) - E_{\boldsymbol{X}Y}V_q\left(Y\tau(\boldsymbol{X})\right)\right| \le & \quad E_{\boldsymbol{X}}|f_\delta(\boldsymbol{X}) - \tau(\boldsymbol{X})| \\ = & \quad E_{\{\boldsymbol{X}:\tau(\boldsymbol{X})\neq f_\delta(\boldsymbol{X})\}}|f_\delta(\boldsymbol{X}) - \tau(\boldsymbol{X})| \\ \le & \quad \frac{2q}{q+1}\cdot\frac{\epsilon(q+1)}{6q} = \epsilon/3,\end{aligned}$$

where the first inequality comes from the fact that $V_q(u)$ is Lipschitz continuous, i.e.,

$$|V_q(u_1) - V_q(u_2)| \leq |u_1 - u_2|, \ \forall u_1, u_2 \in \mathbb{R}.$$

Notice that $\tau(\boldsymbol{X})$ is also continuous. The definition of the universal kernel implies the existence of a function $f_\epsilon \in \mathcal{H}_K$ such that

$$\left| E_{\boldsymbol{X}Y} V_q \left(Y f_\epsilon(\boldsymbol{X}) \right) - E_{\boldsymbol{X}Y} V_q \left(Y \tau(\boldsymbol{X}) \right) \right| < \sup_{\boldsymbol{X}} |f_\epsilon(\boldsymbol{X}) - \tau(\boldsymbol{X})| < \epsilon/3. \tag{A.8}$$

By combining (A.7), (A.8), and (A.8) we obtain (A.6).

Part (2). In this part we bound the estimation error $\varepsilon_E(\hat{f}_n)$. The proof of the estimation error approaching zero essentially only requires the loss function to be Lipschitz continuous, which holds for the DWD case. Note that RKHS has the following reproducing property (Wahba, 1990; Hastie et al., 2009):

$$\begin{aligned} \langle K(\boldsymbol{x}_i, \boldsymbol{x}), f(\boldsymbol{x}) \rangle_{\mathcal{H}_K} &= f(\boldsymbol{x}_i), \\ \langle K(\boldsymbol{x}_i, \boldsymbol{x}), K(\boldsymbol{x}_j, \boldsymbol{x}) \rangle_{\mathcal{H}_K} &= K(\boldsymbol{x}_i, \boldsymbol{x}_j). \end{aligned}$$

Fix any $\epsilon > 0$. By the KKT condition of (2.15) and the representer theorem, we have

$$\frac{1}{n} \sum_{i=1}^{n} V_q' \left(y_i \hat{f}_n(\boldsymbol{x}_i) \right) y_i K(\boldsymbol{x}_i, \boldsymbol{x}) + 2\lambda_n \hat{f}_n(\boldsymbol{x}) = 0. \tag{A.9}$$

We define $\hat{f}^{[k]}$ as the solution of (2.15) when the kth datum is excluded from training data:

$$\hat{f}^{[k]} = \underset{f \in \mathcal{H}_K}{\operatorname{argmin}} \left[\frac{1}{n} \sum_{i=1, i \neq k}^{n} V_q \left(y_i (f(\boldsymbol{x}_i)) + \lambda_n ||f||_{\mathcal{H}_K}^2 \right] \right.$$

By the definition of $\hat{f}^{[k]}$ and the convexity of V_q, we have

$$\begin{aligned} 0 \leq & \frac{1}{n} \sum_{i=1, i \neq k}^{n} V_q \left(y_i \hat{f}_n(\boldsymbol{x}_i) \right) + \lambda_n ||\hat{f}_n||_{\mathcal{H}_K}^2 - \frac{1}{n} \sum_{i=1, i \neq k}^{n} V_q \left(y_i \hat{f}^{[k]}(\boldsymbol{x}_i) \right) - \lambda_n ||\hat{f}^{[k]}||_{\mathcal{H}_K}^2 \\ \leq & - \frac{1}{n} \sum_{i=1, i \neq k}^{n} V_q' \left(y_i \hat{f}_n(\boldsymbol{x}_i) \right) y_i \left(\hat{f}^{[k]}(\boldsymbol{x}_i) - \hat{f}_n(\boldsymbol{x}_i) \right) + \lambda_n ||\hat{f}_n||_{\mathcal{H}_K}^2 - \lambda_n ||\hat{f}^{[k]}||_{\mathcal{H}_K}^2. \end{aligned}$$

By the reproducing property, we further have

$$
\begin{aligned}
0 \leq & -\frac{1}{n}\sum_{i=1,i\neq k}^{n} V_q'\left(y_i\hat{f}_n(\boldsymbol{x}_i)\right) y_i \left\langle K(\boldsymbol{x}_i,\boldsymbol{x}), \hat{f}^{[k]}(\boldsymbol{x}) - \hat{f}_n(\boldsymbol{x})\right\rangle_{\mathcal{H}_K} + \lambda_n||\hat{f}_n||^2_{\mathcal{H}_K} - \lambda_n||\hat{f}^{[k]}||^2_{\mathcal{H}_K} \\
= & -\frac{1}{n}\sum_{i=1,i\neq k}^{n} V_q'\left(y_i\hat{f}_n(\boldsymbol{x}_i)\right) y_i \left\langle K(\boldsymbol{x}_i,\boldsymbol{x}), \hat{f}^{[k]}(\boldsymbol{x}) - \hat{f}_n(\boldsymbol{x})\right\rangle_{\mathcal{H}_K} \\
& - 2\lambda_n \left\langle \hat{f}_n(\boldsymbol{x}), \hat{f}^{[k]}(\boldsymbol{x}) - \hat{f}_n(\boldsymbol{x})\right\rangle_{\mathcal{H}_K} - \lambda_n||\hat{f}^{[k]} - \hat{f}_n||^2_{\mathcal{H}_K} \\
= & \frac{1}{n} V_q'\left(y_k\hat{f}_n(\boldsymbol{x}_k)\right) y_k \left\langle K(\boldsymbol{x}_k,\boldsymbol{x}), \hat{f}^{[k]}(\boldsymbol{x}) - \hat{f}_n(\boldsymbol{x})\right\rangle_{\mathcal{H}_K} - \lambda_n||\hat{f}^{[k]} - \hat{f}_n||^2_{\mathcal{H}_K},
\end{aligned}
$$

where the equality in the end holds by (A.9). Thus, by Cauchy-Schwartz inequality,

$$
\begin{aligned}
& n\lambda_n||\hat{f}^{[k]} - \hat{f}_n||^2_{\mathcal{H}_K} \leq V_q'\left(y_k\hat{f}_n(\boldsymbol{x}_k)\right) y_k \left\langle K(\boldsymbol{x}_k,\boldsymbol{x}), \hat{f}^{[k]}(\boldsymbol{x}) - \hat{f}_n(\boldsymbol{x})\right\rangle_{\mathcal{H}_K} \\
\leq & \left|V_q'\left(y_k\hat{f}_n(\boldsymbol{x}_k)\right)\right| ||K(\boldsymbol{x}_k,\boldsymbol{x})||_{\mathcal{H}_K}||\hat{f}^{[k]} - \hat{f}_n||_{\mathcal{H}_K} \leq \sqrt{K(\boldsymbol{x}_k,\boldsymbol{x}_k)}\cdot||\hat{f}^{[k]} - \hat{f}_n||_{\mathcal{H}_K},
\end{aligned}
$$

which implies

$$
||\hat{f}^{[k]} - \hat{f}_n||_{\mathcal{H}_K} \leq \frac{\sqrt{B}}{n\lambda_n},
$$

where $B = \sup_{\boldsymbol{x}} K(\boldsymbol{x},\boldsymbol{x})$. By the reproducing property, we have

$$
\begin{aligned}
|\hat{f}^{[k]}(\boldsymbol{x}_k) - \hat{f}_n(\boldsymbol{x}_k)|^2 &= \left(\langle K(\boldsymbol{x},\boldsymbol{x}_k), \hat{f}^{[k]}(\boldsymbol{x}) - \hat{f}_n(\boldsymbol{x})\rangle_{\mathcal{H}_K}\right)^2 \\
&\leq K(\boldsymbol{x}_k,\boldsymbol{x}_k)||\hat{f}^{[k]} - \hat{f}_n||^2_{\mathcal{H}_K} \leq B\left(\frac{\sqrt{B}}{n\lambda_n}\right)^2.
\end{aligned}
$$

By the Lipschitz continuity of the DWD loss, we obtain that for each $k = 1,\ldots,n$,

$$
V_q\left(y_k\hat{f}^{[k]}(\boldsymbol{x}_k)\right) - V_q\left(y_k\hat{f}_n(\boldsymbol{x}_k)\right) \leq \ |\hat{f}^{[k]}(\boldsymbol{x}_k) - \hat{f}_n(\boldsymbol{x}_k)| \leq \frac{B}{n\lambda_n},
$$

and therefore,

$$
\frac{1}{n}\sum_{k=1}^{n} V_q\left(y_k\hat{f}^{[k]}(\boldsymbol{x}_k)\right) \leq \frac{1}{n}\sum_{k=1}^{n} V_q\left(y_k\hat{f}_n(\boldsymbol{x}_k)\right) + \frac{B}{n\lambda_n}. \tag{A.10}
$$

Let $f_\epsilon^* \in \mathcal{H}_K$ such that

$$E_{\boldsymbol{X}Y} V_q\left(Y f_\epsilon^*(\boldsymbol{X})\right) \leq \inf_{f \in \mathcal{H}_K} E_{\boldsymbol{X}Y} V_q\left(Y f(\boldsymbol{X})\right) + \epsilon/3.$$

By definition of $\hat{f}_n$, we have

$$\frac{1}{n}\sum_{k=1}^{n} V_q\left(y_k \hat{f}_n(\boldsymbol{x}_k)\right) + \lambda_n ||\hat{f}_n||_{\mathcal{H}_K}^2 \leq \frac{1}{n}\sum_{k=1}^{n} V_q\left(y_k f_\epsilon^*(\boldsymbol{x}_k)\right) + \lambda_n ||f_\epsilon^*||_{\mathcal{H}_K}^2.$$

Since each data point in $\boldsymbol{T}_n = \{(\boldsymbol{x}_k, y_k)\}_{k=1}^n$ is drawn from the same distribution, we have

$$E_{\boldsymbol{T}_n}\left[\frac{1}{n}\sum_{k=1}^{n} V_q\left(y_k \hat{f}^{[k]}(\boldsymbol{x}_k)\right)\right] = \frac{1}{n}\sum_{k=1}^{n} E_{\boldsymbol{T}_n} V_q\left(y_k \hat{f}^{[k]}(\boldsymbol{x}_k)\right) = E_{\boldsymbol{T}_{n-1}} E_{\boldsymbol{X}Y} V_q\left(Y \hat{f}_{n-1}(\boldsymbol{X})\right). \tag{A.11}$$

By combining (A.10)–(A.11) we have

$$E_{\boldsymbol{T}_{n-1}} E_{\boldsymbol{X}Y} V_q\left(Y \hat{f}_{n-1}(\boldsymbol{X})\right) \leq \inf_{f \in \mathcal{H}_K} E_{\boldsymbol{X}Y} V_q\left(Y f(\boldsymbol{X})\right) + \lambda_n ||f_\epsilon^*||_{\mathcal{H}_K}^2 + \frac{B}{n\lambda_n} + \frac{\epsilon}{3}.$$

By the choice of λ_n, we see that there exits N_ϵ such that when $n > N_\epsilon$ we have $\lambda_n < \epsilon/(3||f_\epsilon^*||_{\mathcal{H}_K}^2)$, $n\lambda_n > 3B/\epsilon$, and hence

$$E_{\boldsymbol{T}_{n-1}}\left[E_{\boldsymbol{X}Y} V_q\left(Y \hat{f}_{n-1}(\boldsymbol{X})\right)\right] \leq \inf_{f \in \mathcal{H}_K} E_{\boldsymbol{X}Y} V_q\left(Y f(\boldsymbol{X})\right) + \epsilon.$$

Because ϵ is arbitrary and $E_{\boldsymbol{T}_{n-1}}[E_{\boldsymbol{X}Y} V_q(Y \hat{f}_{n-1}(\boldsymbol{X}))] \geq \inf_{f \in \mathcal{H}_K} E_{\boldsymbol{X}Y} V_q\left(Y f(\boldsymbol{X})\right)$, we have $\lim_{n\to\infty} E_{\boldsymbol{T}_{n-1}}[E_{\boldsymbol{X}Y} V_q(Y \hat{f}_{n-1}(\boldsymbol{X}))] = \inf_{f \in \mathcal{H}_K} E_{\boldsymbol{X}Y} V_q\left(Y f(\boldsymbol{X})\right)$, which equivalently indicates that $\lim_{n\to\infty} E_{\boldsymbol{T}_n} \varepsilon_E(\hat{f}_n) = 0$. Since $\varepsilon_E(\hat{f}_n) \geq 0$, then by Markov inequality we prove part (2).

Appendix B

Proof of Chapter 3

Proof of Lemma 3.1

(3.9) is trivial. To prove (3.10), it suffices to show for any $a \neq b \in \mathbb{R}$,

$$V(a) < V(b) + V'(b)(a-b) + 2(a-b)^2. \tag{B.1}$$

First, it is not hard to check that the first-order derivative $V'(\cdot)$ is Lipschitz continuous, i.e., for any $a \neq b$,

$$|V'(a) - V'(b)| < 4|a-b|. \tag{B.2}$$

Let $g(a) = 2a^2 - V(a)$, then (B.2) shows $g'(a) \equiv 4a - V'(a)$ is strictly increasing. Therefore $g(a)$ is a strictly convex function, and its first-order condition leads to (B.1) directly.

Appendix C

Proof of Chapter 4

Proof of Theorem (4.8)

For simplicity we write $p_j = p_j(\boldsymbol{x})$. Using the Lagrangian multiplier method, we define

$$L(\boldsymbol{f}) = p_1\phi(f_1) + \ldots + p_k\phi(f_k) + \mu(f_1 + \ldots + f_k).$$

Then for each $j = 1, \ldots, k$,

$$\partial L(f)/\partial f_j = \phi'(f_j)p_j + \mu = 0,$$

where $\phi'(f_j) = -1$ if $f_j \leq 1/2$ or $\phi'(f_j) = -1/(4f_j^2)$ if $f_j > 1/2$. We notice that $-1 \leq \phi' < 0$.

Without loss of generality, assume $p_1 > p_2 \geq p_3 \geq \ldots \geq p_{k-1} > p_k$. We observe that $\mu \leq p_j$, $\forall j$. If assume that $\mu < p_k < p_j$, then for any $1 \leq j \leq k$, $\phi'(f_j) = -1/(4f_j^2)$, which gives that

$$f_j = \frac{1}{2}\sqrt{\frac{p_j}{\mu}} > 0, \tag{C.1}$$

contradicting the fact that $\sum_{j=1}^{k} f_j = 0$, so $\mu = p_k$ and $\mu < p_j$ for any $j < k$. Hence, by considering the constraint $\sum_{j=1}^{k} f_j = 0$ and using the same argument as obtaining the

solution (C.1), we have that

$$f_j^\star = \begin{cases} \frac{1}{2}\sqrt{\frac{p_j}{p_k}}, & j < k, \\ -\frac{1}{2}\sum_{j=1}^{k-1}\sqrt{\frac{p_j}{p_k}}, & j = k. \end{cases}$$

Proof of Theorem 4.2

Consider any feasible solution $\boldsymbol{f}$. For each $j = 1, \ldots, k$, we write

$$f_j(\boldsymbol{x}) = \sum_{i=1}^{n} \alpha_{ij} K(\boldsymbol{x}_i, \boldsymbol{x}) + \rho_j(\boldsymbol{x})$$

where ρ_j is orthogonal to the span of $\{K(\boldsymbol{x}_i, \boldsymbol{x})\}$. By the sum-to-zero constaint, we have

$$\sum_{j=1}^{k} \left[\sum_{i=1}^{n} \alpha_{ij} K(\boldsymbol{x}_i, \boldsymbol{x}) + \rho_j(\boldsymbol{x}) \right] = 0$$

or equivalently

$$\left[\sum_{i=1}^{n} \left(\sum_{j=1}^{k} \alpha_{ij} \right) K(\boldsymbol{x}_i, \boldsymbol{x}) \right] + \left[\sum_{j=1}^{k} \rho_j(\boldsymbol{x}) \right] = 0.$$

Since $\sum_{j=1}^{k} \rho_j(\boldsymbol{x})$ is orthogonal to the span of $\{K(\boldsymbol{x}_i, \boldsymbol{x})\}$ and K is a positive definite kernel, the above identity holds if and only if

$$\sum_{j=1}^{k} \alpha_{ij} = 0$$

and

$$\sum_{j=1}^{k} \rho_j(\boldsymbol{x}) = 0.$$

Define $g_j(\boldsymbol{x}) = \sum_{i=1}^{n} \alpha_{ij} K(\boldsymbol{x}_i, \boldsymbol{x})$. So we can write $f_j = g_j + \rho_j$ and $\sum_{j=1}^{k} g_j(\boldsymbol{x}) = 0$, which means $(g_1, \ldots, g_k)$ is another feasible solution.

By the orthogonality of ρ_j and g_j, we have $\|f_j\|_{\mathcal{H}_K}^2 = \|g_j\|_{\mathcal{H}_K}^2 + \|\rho_j\|_{\mathcal{H}_K}^2$. On the other hand, we show $g_j(\boldsymbol{x}_i) = f_j(\boldsymbol{x}_i)$ for all i. For that, we use the reproducing property (Wahba,

1990) and have $\rho_j(\boldsymbol{x}_i) = \langle \rho_j(x), K(\boldsymbol{x}_i, \boldsymbol{x})\rangle_{\mathcal{H}_K} = 0$.

To sum up, for every feasible solution $\boldsymbol{f}$, we can find a better (or at least no worse) feasible solution $\boldsymbol{g}$ such that the two feasible solutions have the same empirical loss value but the latter also has a smaller penalty term. The two feasible solutions are identical if and only if $\rho_j = 0$ for all j. This completes the proof.

Proof of Proposition 4.1

Let ϕ be the DWD loss (4.3). We observe that for any $u_1 \neq u_2$,

$$|\phi'(u_1) - \phi'(u_2)| < 4|u_1 - u_2|.$$

Therefore, we see that

$$\begin{aligned}
&\|\nabla F(\boldsymbol{\alpha}) - \nabla F(\boldsymbol{\alpha}')\| \\
\leq &\left\| \frac{1}{n} \sum_{i=1}^{n} \left(\phi' \left\{ \boldsymbol{K}_i^\top (\boldsymbol{e}_{y_i}^\top \otimes \boldsymbol{I}_n) \boldsymbol{\alpha} \right\} - \phi' \left\{ \boldsymbol{K}_i^\top (\boldsymbol{e}_{y_i}^\top \otimes \boldsymbol{I}_n) \boldsymbol{\alpha}' \right\} \right) (\boldsymbol{e}_{y_i} \otimes \boldsymbol{I}_n) \boldsymbol{K}_i + 2\lambda (\boldsymbol{I}_k \otimes \boldsymbol{K}) (\boldsymbol{\alpha} - \boldsymbol{\alpha}') \right\| \\
\leq &\left\| \left(\frac{4}{n} \sum_{i=1}^{n} (\boldsymbol{e}_{y_i} \otimes \boldsymbol{I}_n) \boldsymbol{K}_i \boldsymbol{K}_i^\top (\boldsymbol{e}_{y_i}^\top \otimes \boldsymbol{I}_n) + 2\lambda (\boldsymbol{I}_k \otimes \boldsymbol{K}) \right) (\boldsymbol{\alpha} - \boldsymbol{\alpha}') \right\| \\
\leq &L(F) \left\| (\boldsymbol{\alpha} - \boldsymbol{\alpha}') \right\|,
\end{aligned} \tag{C.2}$$

where $L(F)$ is the largest eigenvalue of the matrix

$$\boldsymbol{T} \equiv \frac{4}{n} \sum_{i=1}^{n} (\boldsymbol{e}_{y_i} \otimes \boldsymbol{I}_n) \boldsymbol{K}_i \boldsymbol{K}_i^\top (\boldsymbol{e}_{y_i}^\top \otimes \boldsymbol{I}_n) + 2\lambda (\boldsymbol{I}_k \otimes \boldsymbol{K}),$$

which is an $nk \times nk$ block diagonal matrix whose j^{th} block is an $n \times n$ matrix

$$\boldsymbol{T}_j = \frac{4}{n} \sum_{\{i: y_i = j\}} \boldsymbol{K}_i \boldsymbol{K}_i^\top + 2\lambda \boldsymbol{K}.$$

The largest eigenvalue of $\boldsymbol{T}_j$ is $4\tilde{\sigma}_j/n + 2\lambda\sigma$, then $L(F)$, the largest eigenvalue of $\boldsymbol{T}$, is $4\tilde{\sigma}/n + 2\lambda\sigma$. The proposition is then proved by applying theorem 3.1 of Beck and Teboulle (2009).

Proof of Proposition 4.2

The proposition follows inequality (C.2) and theorem 4.4 of Beck and Teboulle (2009).

Appendix D

Proof of Chapter 5

Proof of Lemma 5.1

By the definition of $\tilde{\boldsymbol{y}}$ and $\hat{f}^{[-v]}$, for any function f, we have that,

$$\begin{aligned}\frac{1}{n}\sum_{i=1}^{n}\left(\tilde{y}_i - \hat{f}_\lambda^{[-v]}(\boldsymbol{x}_i)\right)^2 + \lambda P(\hat{f}_\lambda^{[-v]}) =& \frac{1}{n}\sum_{i=1,\ i\neq v}^{n}\left(y_i - \hat{f}_\lambda^{[-v]}(\boldsymbol{x}_i)\right)^2 + \lambda P(\hat{f}_\lambda^{[-v]}) \\ \leq& \frac{1}{n}\sum_{i=1,\ i\neq v}^{n}\left(y_i - f(\boldsymbol{x}_i)\right)^2 + \lambda P(f) \\ \leq& \frac{1}{n}\sum_{i=1}^{n}\left(\tilde{y}_i - f(\boldsymbol{x}_i)\right)^2 + \lambda P(f).\end{aligned}$$

Then we see that $\hat{f}_\lambda^{[-v]}(\boldsymbol{x}_i) = \boldsymbol{H}_i^T\tilde{\boldsymbol{y}}$, and thus

$$y_i - \hat{f}_\lambda(\boldsymbol{x}_i) = y_i - \boldsymbol{H}_i^T\boldsymbol{y} = y_i - \boldsymbol{H}_i^T\tilde{\boldsymbol{y}} - h_{ii}(y_i - \boldsymbol{H}_i^T\tilde{\boldsymbol{y}}) = (1-h_{ii})(y_i - \hat{f}_\lambda^{[-v]}(\boldsymbol{x}_i)).$$

The conclusion follows.

Proof of Theorem 5.1

The KKT conditions of (5.9) indicate that

$$\frac{1}{n}\sum_{i=1}^{n} y_i L'^{\delta}(\theta_i^\delta) + 2\lambda\boldsymbol{K}^{-1}\boldsymbol{\theta}^\delta = \boldsymbol{0}, \tag{D.1}$$

where each $L'^{\delta}(\theta_i^{\delta}) \in [-1, 0]$. Define $E_0 = \{i : y_i\theta_i^{\mathrm{SVM}} = 1\}$. Let $\delta_1 = \min_{i \notin E_0}\{|1 - y_i\theta_i^{\mathrm{SVM}}|\}$. Define $\mathcal{C}(\boldsymbol{\theta})$ to be a set $\left\{\boldsymbol{\theta} : \|\boldsymbol{\theta}^{\delta} - \boldsymbol{\theta}^{\mathrm{SVM}}\|_{\infty} \geq \delta_1/2\right\}$.

Let $\eta = \inf_{\boldsymbol{\theta} \in \mathcal{C}(\boldsymbol{\theta})}[Q(\boldsymbol{\theta}) - Q(\boldsymbol{\theta}^{\mathrm{SVM}})] > 0$. Since $Q^{\delta}(\boldsymbol{\theta})$ uniformly converges to $Q(\boldsymbol{\theta})$, there exists δ_2 such that for any $\delta < \delta_2$,

$$\begin{aligned} Q^{\delta}(\boldsymbol{\theta}^{\mathrm{SVM}}) - Q(\boldsymbol{\theta}^{\mathrm{SVM}}) &< \eta/2, \\ Q(\boldsymbol{\theta}^{\delta}) - Q^{\delta}(\boldsymbol{\theta}^{\delta}) &< \eta/2. \end{aligned}$$

Since $Q^{\delta}(\boldsymbol{\theta}^{\delta}) < Q^{\delta}(\boldsymbol{\theta}^{\mathrm{SVM}})$, we have $Q(\boldsymbol{\theta}^{\delta}) < Q(\boldsymbol{\theta}^{\mathrm{SVM}}) + \eta$, which follows $\boldsymbol{\theta}^{\delta} \notin \mathcal{C}(\boldsymbol{\theta})$. Therefore,

$$|y_i\theta_i^{\mathrm{SVM}} - y_i\theta_i^{\delta}| < \delta_1/2, \ \forall i. \tag{D.2}$$

Let $\delta^{\star} = \min\{\delta_1/2, \delta_2\}$. For any $\delta < \delta^{\star}$, define $E_0^{\delta} = \{i : |1 - y_i\theta_i^{\delta}| < \delta\}$. For any $i \in E_0^{\delta}$, the definition of E_0^{δ} and (D.2) yield

$$|1 - y_i\theta_i^{\mathrm{SVM}}| \leq |1 - y_i\theta_i^{\delta}| + |y_i\theta_i^{\delta} - y_i\theta_i^{\mathrm{SVM}}| < \delta + \delta_1/2 < \delta_1,$$

indicating $i \in E_0$ and thus $E_0^{\delta} \subseteq E_0$. Accordingly, for any $i \notin E_0$, then $i \notin E_0^{\delta}$, hence

$$L'^{\delta}(\theta_i^{\delta}) = L'(\theta_i^{\delta}) = \begin{cases} 0 & t > 1 + \delta_1, \\ -1 & t < 1 + \delta_1. \end{cases}$$

For each $i \in E_0$, define $\xi_i = L'^{\delta}(\theta_i^{\delta}) \in [-1, 0]$. Then from (D.1) we have that

$$\begin{aligned} 0 &= \frac{1}{n}\sum_{i \in E_0} y_i L'^{\delta}(\theta_i^{\delta}) + \frac{1}{n}\sum_{i \notin E_0} y_i L'^{\delta}(\theta_i^{\delta}) + 2\lambda \boldsymbol{K}^{-1}\boldsymbol{\theta}^{\delta} \\ &= \frac{1}{n}\sum_{i \in E_0} y_i \xi_i + \frac{1}{n}\sum_{i \notin E_0} y_i L'^{\delta}(\theta_i^{\delta}) + 2\lambda \boldsymbol{K}^{-1}\boldsymbol{\theta}^{\delta} \\ &= \frac{1}{n}\sum_{i \in E_0} y_i \xi_i + \frac{1}{n}\sum_{i \notin E_0} y_i L'(\theta_i^{\delta}) + 2\lambda \boldsymbol{K}^{-1}\boldsymbol{\theta}^{\delta}. \end{aligned}$$

Therefore $\boldsymbol{\theta}^{\delta}$ satisfies the KKT condition of (5.8). This demonstrates that $\boldsymbol{\theta}^{\delta}$ is the solution of (5.8) for any $\delta < \delta^{\star}$ and hence completes the proof.