ROYAL STATISTICAL SOCIETY
DATA | EVIDENCE | DECISIONS

Journal of the Royal Statistical Society
Statistical Methodology
Series B

J. R. Statist. Soc. B (2018)
80, *Part* 1, *pp.* 177–198

Another look at distance-weighted discrimination

Boxiang Wang and Hui Zou

University of Minnesota, Minneapolis, USA

[Received February 2016. Final revision June 2017]

Summary. 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. 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 state of the art algorithm based on 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.

Keywords: Bayes risk consistency; Classification; Distance-weighted discrimination; Kernel learning; Majorization–minimization principle; Second-order cone programming

1. Introduction

Binary classification problems appear from diverse practical applications, such as financial fraud detection, spam e-mail classification, medical diagnosis with genomics data and drug response modelling, among many others. In these classification problems, the goal is to predict class labels on the basis of a given set of variables. Suppose that we observe a training data set consisting of n pairs, where $\{(\mathbf{x}_i, y_i)\}_{i=1}^n$, $\mathbf{x}_i \in \mathbb{R}^p$, and $y_i \in \{-1, 1\}$. A classifier fits a discriminant function f and constructs a classification rule to classify data point $\mathbf{x}_i$ to either class 1 or class -1 according to the sign of $f(\mathbf{x}_i)$. The decision boundary is given by $\{\mathbf{x} : f(\mathbf{x}) = 0\}$. 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 (SVMs) or kernel machines. Ensemble learning such as boosting (Freund and Schapire, 1997) and random forests (Breiman, 2001) combine many weak learners like decision trees into a powerful learner. The 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

Address for correspondence: Hui Zou, School of Statistics, University of Minnesota, 224 Church Street South East, Minneapolis, MN 55455, USA.
E-mail: zouxx019@umn.edu

 1369–7412/18/80177

'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 state of the art 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 with the kernel SVM. Although Marron *et al.* (2007) proposed a version of non-linear DWD by mimicking the kernel trick that is used for deriving the kernel SVM, theoretical justification of such a kernel DWD is still absent. In contrast, the kernel SVM as well as kernel logistic regression (Wahba *et al.*, 1994; Zhu and Hastie, 2005) have mature theoretical understandings built on the theory of reproducing kernel Hilbert spaces (RKHSs) (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 paper, 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 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 generalized DWD and conducting efficient tuning. We have implemented our algorithms in an R package `kerndwd`, which is publicly available on the Comprehensive R Archive Network at `http://cran.r-project.org/web/packages/kerndwd/index.html`.

The rest of the paper is organized as follows. To be self-contained, we first review the SVM and DWD in Section 2. We then derive a novel algorithm for DWD and generalized DWD in Section 3. We introduce kernel DWD in an RKHS and establish its Bayes risk consistency in Section 4. Numeric examples are given in Section 5, and technical proofs are provided in Appendix A.

2. Review of the support vector machine and distance-weighted discrimination

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 $\{\mathbf{x}: f(x) = \omega_0 + \mathbf{x}^{\mathrm{T}}\boldsymbol{\omega} = 0\}$ where $y_i(\omega_0 + \mathbf{x}_i^{\mathrm{T}}\boldsymbol{\omega})$ are all non-negative. Without loss of generality, we assume that $\boldsymbol{\omega}$ is a unit vector, i.e. $\boldsymbol{\omega}^{\mathrm{T}}\boldsymbol{\omega} = 1$, and we observe that each $d_i \equiv y_i(\omega_0 + \mathbf{x}_i^{\mathrm{T}}\boldsymbol{\omega})$ is equivalent to the Euclidean distance between the data point $\mathbf{x}_i$ and the hyperplane. The reason is that $d_i = (\mathbf{x}_i - \mathbf{x}_0)^{\mathrm{T}}\boldsymbol{\omega}$ and $\omega_0 + \mathbf{x}_0^{\mathrm{T}}\boldsymbol{\omega} = 0$, where $\mathbf{x}_0$ 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 $y_i(\omega_0 + \mathbf{x}_i^{\mathrm{T}}\boldsymbol{\omega}) + \eta_i$ are non-negative. So, the SVM is defined as

$$\max_{\omega_0,\boldsymbol{\omega},d_i,\eta_i} \min d_i, \qquad \text{subject to } d_i = y_i(\omega_0 + \mathbf{x}_i^{\mathrm{T}}\boldsymbol{\omega}) + \eta_i \geqslant 0,\ \forall i,$$
$$\eta_i \geqslant 0,\ \forall i,\ \sum_{i=1}^{n} \eta_i < \text{ constant, and } \boldsymbol{\omega}^{\mathrm{T}}\boldsymbol{\omega} = 1. \tag{2.1}$$

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:

$$\min_{\omega_0,\boldsymbol{\omega},d_i,\eta_i} \left(\sum_{i=1}^{n} \frac{1}{d_i} + c \sum_{i=1}^{n} \eta_i \right), \qquad \text{subject to } d_i = y_i(\omega_0 + \mathbf{x}_i^{\mathrm{T}}\boldsymbol{\omega}) + \eta_i \geqslant 0,\ \eta_i \geqslant 0,\ \forall i, \text{ and } \boldsymbol{\omega}^{\mathrm{T}}\boldsymbol{\omega} = 1. \tag{2.2}$$

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 an SOCP programme (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

$$\min_{\omega_0,\boldsymbol{\omega},\rho_i,\sigma_i,\eta_i} \left\{ \sum_{i=1}^{n} (\rho_i + \sigma_i) + c \sum_{i=1}^{n} \eta_i \right\}, \qquad \text{subject to } \rho_i - \sigma_i = y_i \mathbf{x}_i^{\mathrm{T}}\boldsymbol{\omega} + \omega_0 y_i + \eta_i,\ \forall i,$$
$$\eta_i \geqslant 0,\ (\rho_i; \sigma_i, 1) \in S_3,\ \forall i,\ (1; \boldsymbol{\omega}) \in S_{p+1}, \tag{2.3}$$

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

There has been much work on variants of 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. Wang and Zou (2016) proposed sparse DWD for high dimensional classification. In addition, the work connecting DWD with other classifiers, e.g. the SVM, includes but is not limited to large margin unified machines (Liu *et al.*, 2011), the distance-weighted SVM (Qiao and Zhang, 2015a) and the flexible assortment machine (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 DWD with this new formulation generalized DWD:

$$\min_{\omega_0,\boldsymbol{\omega}} \left(\sum_{i=1}^{n} \frac{1}{d_i^q} + c \sum_{i=1}^{n} \eta_i \right), \qquad \text{subject to } d_i = y_i(\omega_0 + \mathbf{x}_i^{\mathrm{T}}\boldsymbol{\omega}) + \eta_i \geqslant 0,\ \eta_i \geqslant 0,\ \forall i, \text{ and } \boldsymbol{\omega}^{\mathrm{T}}\boldsymbol{\omega} = 1, \tag{2.4}$$

which degenerates to the standard DWD (2.2) when $q = 1$. Generalized DWD has not been implemented yet because the SOCP transformation works only 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 on DWD with $q = 1$ only. 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 paper can easily solve the generalized DWD problem for any $q > 0$; see Section 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. $\mathbf{K} = \mathbf{\Phi}\mathbf{\Phi}^{\mathrm{T}}$ and then replaced the design matrix $\mathbf{X}$ with $\mathbf{\Phi}$. To our best knowledge, still unclear is the theoretical justification for kernel DWD in Marron *et al.* (2007). The reason is likely to be that the non-linear extension is purely algorithmic. Kernel DWD considered in this paper can be rigorously justified to have a universal Bayes risk consistency property; see the details in Section 4.2.

3. A novel algorithm for distance-weighted discrimination and generalized distance-weighted discrimination

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

3.1. Generalized distance-weighted discrimination loss

Our algorithm begins with a *loss-plus-penalty* formulation of DWD. Lemma 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 4.

Lemma 1. The generalized DWD classifier (2.4) can be written as $\mathrm{sgn}(\hat{\beta}_0 + \mathbf{x}_i^{\mathrm{T}}\hat{\boldsymbol{\beta}})$, where $(\hat{\beta}_0, \hat{\boldsymbol{\beta}})$ is computed for some λ from

$$\min_{\beta_0,\boldsymbol{\beta}} \mathbf{C}(\beta_0, \boldsymbol{\beta}) \equiv \min_{\beta_0,\boldsymbol{\beta}} \left[\frac{1}{n} \sum_{i=1}^{n} V_q\{y_i(\beta_0 + \mathbf{x}_i^{\mathrm{T}}\boldsymbol{\beta})\} + \lambda \boldsymbol{\beta}^{\mathrm{T}}\boldsymbol{\beta} \right]; \tag{3.1}$$

$$V_q(u) = \begin{cases} 1-u, & \text{if } u \leqslant \dfrac{q}{q+1}, \\ \dfrac{1}{u^q}\dfrac{q^q}{(q+1)^{q+1}}, & \text{if } u > \dfrac{q}{q+1}. \end{cases} \tag{3.2}$$

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

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

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

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) and Lange and Zhou (2014), among others. The main idea of the MM principle is easy to understand. Suppose that $\boldsymbol{\theta} = (\beta_0, \boldsymbol{\beta}^{\mathrm{T}})^{\mathrm{T}}$ and

we aim to minimize $\mathbf{C}(\boldsymbol{\theta})$, defined in expression (3.1). The MM principle finds a majorization function $\mathbf{D}(\boldsymbol{\theta}|\boldsymbol{\theta}_k)$ satisfying $\mathbf{C}(\boldsymbol{\theta}) < \mathbf{D}(\boldsymbol{\theta}|\boldsymbol{\theta}_k)$ for any $\boldsymbol{\theta} \neq \boldsymbol{\theta}_k$ and $\mathbf{C}(\boldsymbol{\theta}_k) = \mathbf{D}(\boldsymbol{\theta}_k|\boldsymbol{\theta}_k)$, and then we generate a sequence $\{\mathbf{C}(\boldsymbol{\theta}_k)\}_{k=1}^{\infty}$ via $\boldsymbol{\theta}_{k+1} = \arg\min_{\boldsymbol{\theta}} \mathbf{D}(\boldsymbol{\theta}|\boldsymbol{\theta}_k)$.

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

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

Lemma 2. The generalized DWD loss function $V_q(\cdot)$ has a Lipschitz continuous gradient,

$$|V_q'(t) - V_q'(\tilde{t})| < \frac{(q+1)^2}{q}|t - \tilde{t}|, \tag{3.3}$$

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

$$V_q(t) < V_q(\tilde{t}) + V_q'(\tilde{t})(t - \tilde{t}) + \frac{(q+1)^2}{2q}(t - \tilde{t})^2. \tag{3.4}$$

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

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

where

$$\tilde{\boldsymbol{\gamma}} \equiv \begin{pmatrix} \mathbf{1}^{\mathrm{T}}\tilde{\mathbf{z}} \\ \mathbf{X}^{\mathrm{T}}\tilde{\mathbf{z}} + 2\lambda\tilde{\boldsymbol{\beta}} \end{pmatrix},$$

$$\mathbf{P}_{q,\lambda} \equiv \begin{pmatrix} n & \mathbf{1}^{\mathrm{T}}\mathbf{X} \\ \mathbf{X}^{\mathrm{T}}\mathbf{1} & \mathbf{X}^{\mathrm{T}}\mathbf{X} + \dfrac{2nq\lambda}{(q+1)^2}\mathbf{I}_{p\times p} \end{pmatrix},$$

and the inequality comes from lemma 2 with the equality held only when $(\beta_0, \boldsymbol{\beta}) = (\tilde{\beta}_0, \tilde{\boldsymbol{\beta}})$. To minimize $\mathbf{D}(\beta_0, \boldsymbol{\beta})$, we set $[\partial\mathbf{D}(\beta_0, \boldsymbol{\beta})/\partial\beta_0, \partial\mathbf{D}(\beta_0, \boldsymbol{\beta})/\partial\boldsymbol{\beta}]$ to be 0s and then we have

$$\begin{pmatrix} \beta_0 \\ \boldsymbol{\beta} \end{pmatrix} = \begin{pmatrix} \tilde{\beta}_0 \\ \tilde{\boldsymbol{\beta}} \end{pmatrix} - \frac{nq}{(q+1)^2}\mathbf{P}_{q,\lambda}^{-1}\tilde{\boldsymbol{\gamma}}. \tag{3.5}$$

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 $\mathbf{P}_{q,\lambda}$ is repeatedly inverted as q and λ are varied. Inverting a large matrix can be costly. Here we exploit the special structure of $\mathbf{P}_{q,\lambda}$ 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 linear DWD with $n < p$, we can treat it as special kernel DWD with a linear kernel and then we need to invert only an $n \times n$ matrix. Our treatment also works for kernel DWD, which will be discussed in Section 4.

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

$$\mathbf{P}_0 \equiv \begin{pmatrix} n & \mathbf{1}^{\mathrm{T}}\mathbf{X} \\ \mathbf{X}^{\mathrm{T}}\mathbf{1} & \mathbf{X}^{\mathrm{T}}\mathbf{X} \end{pmatrix} = \mathbf{U}\mathbf{\Pi}\mathbf{U}^{\mathrm{T}}, \tag{3.6}$$

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

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

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

$$\mathbf{P}_{q,\lambda}^{-1} = \left\{ \mathbf{Q}_{q,\lambda} + \begin{pmatrix} -\dfrac{2nq\lambda}{(q+1)^2} & \mathbf{0}^{\mathrm{T}} \\ \mathbf{0} & \mathbf{0}_{p\times p} \end{pmatrix} \right\}^{-1} = \mathbf{Q}_{q,\lambda}^{-1} + g\mathbf{v}\mathbf{v}^{\mathrm{T}} = \mathbf{U}\mathbf{\Pi}_{q,\lambda}^{-1}\mathbf{U}^{\mathrm{T}} + g\mathbf{v}\mathbf{v}^{\mathrm{T}}, \tag{3.7}$$

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

Finally, we directly compute $\mathbf{P}_{q,\lambda}^{-1}\tilde{\gamma}$ in equation (3.5) rather than $\mathbf{P}_{q,\lambda}^{-1}$. By equation (3.7), we see that

$$\mathbf{P}_{q,\lambda}^{-1}\gamma = (\mathbf{Q}_{q,\lambda}^{-1} + g\mathbf{v}\mathbf{v}^{\mathrm{T}})\tilde{\gamma} = \mathbf{U}\mathbf{\Pi}_{q,\lambda}^{-1}\mathbf{U}^{\mathrm{T}}\tilde{\gamma} + g\mathbf{v}\mathbf{v}^{\mathrm{T}}\tilde{\gamma}. \tag{3.8}$$

Note that $\mathbf{v}$ can be obtained via $\mathbf{U}\mathbf{\Pi}_{q,\lambda}^{-1}\mathbf{u}_1$ where $\mathbf{u}_1$ is the first row of $\mathbf{U}$; hence we find that all the operations in equation (3.8) are $\mathcal{O}(p^2)$. Therefore, we manage to do the actual matrix inversion once in equation (3.6), when solving generalized DWD with different q- and λ-values.

4. Kernel distance-weighted discrimination in reproducing kernel Hilbert space and Bayes risk consistency

4.1. Kernel distance-weighted discrimination in reproducing kernel Hilbert space

The kernel SVM can be derived by using the kernel trick or by using the view of non-parametric function estimation in an 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 that K has an eigenexpansion $K(\mathbf{x}, \mathbf{x}') = \Sigma_{t=1}^{\infty}\gamma_t\phi_t(\mathbf{x})\phi_t^{\mathrm{T}}(\mathbf{x}')$, with $\gamma_t \geqslant 0$ and $\Sigma_{t=1}^{\infty}\gamma_t^2 < \infty$. Then the

Hilbert space $\mathcal{H}_K$ is defined as the collection of functions $h(\mathbf{x}) = \Sigma_{t=1}^{\infty} \theta_t \phi_t(\mathbf{x})$, for any θ_t such that $\Sigma_{t=1}^{\infty} \theta_t^2/\gamma_t < \infty$, and the inner product is $\langle \Sigma_{t=1}^{\infty} \theta_t \phi_t(\mathbf{x}), \Sigma_{t'=1}^{\infty} \delta_{t'} \phi_{t'}(\mathbf{x}) \rangle_{\mathcal{H}_K} = \Sigma_{t=1}^{\infty} \theta_t \delta_t/\gamma_t$.

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

$$\min_{\substack{h \in \mathcal{H}_K \\ \beta_0 \in \mathbb{R}}} \left(\frac{1}{n} \sum_{i=1}^{n} V_q[y_i\{\beta_0 + h(\mathbf{x}_i)\}] + \lambda ||h||^2_{\mathcal{H}_K} \right), \tag{4.1}$$

and $V_q(\cdot)$ is the generalized DWD loss (3.2). The representer theorem concludes that the solution of problem (4.1) has a finite expansion based on $K(\mathbf{x}, \mathbf{x}_i)$ (Wahba, 1990), $\hat{h}(\mathbf{x}) = \Sigma_{i=1}^{n} \hat{\alpha}_i K(\mathbf{x}, \mathbf{x}_i)$, and thus $||\hat{h}||^2_{\mathcal{H}_K} = \Sigma_{i=1}^{n} \Sigma_{j=1}^{n} \hat{\alpha}_i \hat{\alpha}_j K(\mathbf{x}_i, \mathbf{x}_j)$. Consequently, problem (4.1) can be paraphrased with matrix notation:

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

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

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

$$\mathbf{P}_{q,\lambda} \equiv \begin{pmatrix} n & \mathbf{1}^{\mathrm{T}}\mathbf{K} \\ \mathbf{K1} & \mathbf{KK} + \dfrac{2nq\lambda}{(q+1)^2}\mathbf{K} \end{pmatrix}.$$

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

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

whose closed form minimizer is obtained as

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

The direct use of this formula requires computing the inverse of $n \times n$ matrix $\mathbf{P}_{q,\lambda}$ 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 $\mathbf{K} = \mathbf{U}\boldsymbol{\Lambda}\mathbf{U}^{\mathrm{T}}$, which is free of tuning parameters, and we then compute $\boldsymbol{\Pi}_{q,\lambda} = \boldsymbol{\Lambda}\boldsymbol{\Lambda} + \{2nq\lambda/(q+1)^2\}\boldsymbol{\Lambda}$ for each q and λ. With $\mathbf{v} = \mathbf{U}\boldsymbol{\Lambda}\boldsymbol{\Pi}_{q,\lambda}^{-1}\mathbf{U}^{\mathrm{T}}\mathbf{1}$ being computed through $\mathcal{O}(n^2)$ operations and $g = 1/(n - \mathbf{1}^{\mathrm{T}}\mathbf{U}\boldsymbol{\Lambda}\boldsymbol{\Pi}_{q,\lambda}^{-1}\boldsymbol{\Lambda}\mathbf{U}^{\mathrm{T}}\mathbf{1})$, we glean the decomposition

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

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

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

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

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 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 with a nice road map to study kernel DWD. In this section we first elucidate the Fisher consistency (Lin, 2004) of generalized kernel DWD, and we then establish the Bayes risk consistency of kernel DWD when a universal kernel is employed.

Let $\eta(\mathbf{x})$ denote the conditional probability $P(Y=1|\mathbf{X}=\mathbf{x})$. Under 0–1-loss, the theoretical optimal Bayes rule is $f^*(\mathbf{x})=\operatorname{sgn}\{\eta(\mathbf{x})-\frac{1}{2}\}$. Assume that $\eta(\mathbf{x})$ is a measurable function and $P\{\eta(\mathbf{x})=\frac{1}{2}\}=0$ throughout.

Lemma 3. The population minimizer of the expected DWD loss $E_{\mathbf{X}Y}[V_q\{Yf(\mathbf{X})\}]$ is

$$\tilde{f}(\mathbf{x})=\frac{q}{q+1}\left[\left\{\frac{\eta(\mathbf{x})}{1-\eta(\mathbf{x})}\right\}^{1/(q+1)} I\{\eta(\mathbf{x})>\tfrac{1}{2}\}-\left\{\frac{1-\eta(\mathbf{x})}{\eta(\mathbf{x})}\right\}^{1/(q+1)} I\{\eta(\mathbf{x})<\tfrac{1}{2}\}\right]. \tag{4.2}$$

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

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 3 that the generalized DWD loss is Fisher consistent. Lemma 3 is a generalization of a previously shown result (Qiao *et al.*, 2010; Liu *et al.*, 2011) that proves that the standard DWD loss $V_1(u)$ is Fisher consistent. It is worth emphasizing that the condition and conclusion of lemma 3 are independent of the functional space: Fisher consistency hinges on only the loss function. Lemma 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 what follows. Such results are missing in the current DWD literature.

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

$$\hat{f}_n=\arg\min_{f\in\mathcal{H}_K}\left[\frac{1}{n}\sum_{i=1}^{n} V_q\{y_i f(\mathbf{x}_i)\}+\lambda_n\|f\|^2_{\mathcal{H}_K}\right]. \tag{4.3}$$

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 the kernel DWD classifier works as well as the Bayes rule (asymptotically speaking). For this, we present lemma 4, which is closely related to theorem 2.1 of Zhang (2004) and theorem 3 of Bartlett *et al.* (2006).

Lemma 4. For a discrimination function f, we define $R(f)=E_{\mathbf{X}Y}[Y\neq\operatorname{sgn}\{f(\mathbf{X})\}]$. Assume that $f^*=\arg\min_f R(f)$ is the Bayes rule and $\hat{f}_n$ is the solution of equation (4.3); then

$$R(\hat{f}_n)-R(f^*)\leqslant\frac{q+1}{q}(\varepsilon_{\mathrm{A}}+\varepsilon_{\mathrm{E}}), \tag{4.4}$$

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

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

In lemma 4 $R(f^*)$ is the Bayes error rate and $R(\hat{f}_n)$ is the misclassification rate of kernel DWD applied to new data points. If $R(\hat{f}_n) \to R(f^*)$, we say that the classifier is Bayes risk consistent. On the basis of lemma 4, it suffices to show that both ε_A and ε_E approach 0 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 sufficiently rich then the approximation error can be made arbitrarily small. In the literature, the notation of a *universal kernel* (Steinwart, 2001; Micchelli *et al.*, 2006) has been proposed and studied. Suppose that $\mathcal{X} \in \mathbb{R}^p$ is the compact input space of $\mathbf{X}$ 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 $\mathcal{H}_K$ that is generated by K is dense in $C(\mathcal{X})$, i.e., for any positive ϵ and any function $g \in C(\mathcal{X})$, there is an $f \in \mathcal{H}_K$ such that $\|f - g\|_\infty < \epsilon$.

Theorem 1. Suppose that $\hat{f}_n$ is the solution of equation (4.3), $\mathcal{H}_K$ is induced by a universal kernel K and the sample space $\mathcal{X}$ is compact. Then we have

(a) $\varepsilon_A = 0$;
(b) let $B = \sup_{\mathbf{x}} K(\mathbf{x}, \mathbf{x}) < \infty$. When $\lambda_n \to 0$ and $n\lambda_n \to \infty$, for any $\epsilon > 0$,

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

By results (a) and (b) and inequality (4.4) we have $R(\hat{f}_n) \to R(f^*)$ in probability.

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

5. Numeric examples

5.1. Timing comparison with second-order cone programming 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 solve only linear DWD with $q = 1$, we report only the timing of computing 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 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 by using the R packages `kerndwd` and `DWD`, and the MATLAB implementation respectively.

From Table 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 1. Computation time comparisons between the R package kerndwd, the R package DWD (denoted as HUANG) and the MATLAB implementation MARRON†

Example	*Average time (s)*				*Ratio*	
	WZpath	*WZ*	*HUANG*	*MARRON*	$t(HUANG)/t(WZ)$	$t(MARRON)/t(WZ)$
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

†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 solve only the DWD problem on the best λ. In the 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) central processor unit E5-2660 at 2.60 GHz.

5.2. Timing comparison with the symmetric Gauss–Seidel alternating direction method of multipliers

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 semiproximal alternating direction method of multipliers, which is modified from very advanced mathematical programming work (Li *et al.*, 2016; 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 summarizes the timing comparisons between `DWDLarge` and `kerndwd` on 10 benchmark data sets that were obtained from the University of California at Irvine Machine Learning Repository (Lichman, 2013). Among the 10 data sets, 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 a sample size of over a half million. `DWDLarge` implements only linear DWD with $q = 1, 2, 3, 4$.

In Table 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 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 that the 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 that our methods are roughly of 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.

5.3. Comparing the support vector machine and generalized distance-weighted discrimination

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 1s and 150 0s. In each example, the positive class arose from a mixture Gaussian distribution $\Sigma_{k=1}^{10} 0.1 N(\boldsymbol{\mu}_{k+}, 10^2 \mathbf{I})$ with each $\boldsymbol{\mu}_{k+}$ drawn from $N(\boldsymbol{\mu}_+, \mathbf{I})$, and likewise the negative class was assembled by $\Sigma_{k=1}^{10} 0.1 N(\boldsymbol{\mu}_{k-}, 10^2 \mathbf{I})$ with each $\boldsymbol{\mu}_{k-}$ from $N(\boldsymbol{\mu}_-, \mathbf{I})$. For this

Table 2. Computation time of kerndwd and DWDLarge on 10 University of California at Irvine benchmark data sets†

Data	*n*	*p*	*q*	*WZpath (s)*	*WZ (s)*	*LMST (s)*	*q*	*WZpath (s)*	*WZ (s)*	*LMST (s)*
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

†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 by 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) central processor unit E5-2660 at 2.60 GHz.

model the Bayes rule is non-linear 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 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 for the SVM were obtained by the R package `e1071` (Meyer *et al.*, 2015).

Table 3 summarizes the average time for training the classifier and the misclassification rates assessed on an independent test set with a sample size of 10 000. We can make several observations from Table 3. First, none of the kernel DWD methods 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 misclassification rates of the 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 `kernlab` is through 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 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 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 than the SVM. We also tried $n = 2000$ and $n = 3000$; the time of the kernel DWD was 393.60 s and 1131.50 s

Table 3. Misclassification rates and computation time, averaged by 20 runs, under mixture Gaussian distributed data†

Method	*Results for* $n=100$, $p=300$		*Results for* $n=500$, $p=300$		*Results for* $n=900$, $p=300$		*Results for* $n=1300$, $p=300$	
	Error (%)	*Time (s)*	*Error (%)*	*Time (s)*	*Error (%)*	*Time (s)*	*Error (%)*	*Time (s)*
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

†In each example, we used fivefold 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 give the standard errors in parentheses. We displayed all the time that includes fitting models and tuning the parameters with fivefold cross-validations. In each example, the method incurring the lowest error is marked by italics. The Bayes error rate is 18.85%.

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 data set 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 analysed for the kernel SVM by Hsieh *et al.* (2014) and kernel ridge regression by Zhang *et al.* (2015).

5.4. Benchmark data examples

We examined the performance of `kerndwd` on 16 University of California at Irvine 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 set into a training and a test set with a ratio 2:1. We conducted fivefold 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 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 the kernel SVM and kernel DWD with a data-driven q, we see that the latter has a lower misclassification rate in 10 examples and is significantly faster in 14 examples. Standard DWD with $q=1$ is the fastest to compute, but exploring a data-driven q in DWD can lead to a noticeable improvement in the prediction accuracy with affordable extra computing time.

In Table 5, we compare kernel DWD with another four commonly used classifiers: gradient

Table 4. Mean misclassification rates and computation time for the SVM, standard DWD ($q = 1$) and DWD with a data-driven q on 16 benchmark data sets†

Data	*n*	*p*	*Kernel*	*Results for SVM*		*Results for* $DWD_{q=1}$		*Results for* $DWD_{\text{data-driven } q}$	
				Error (%)	*Time (s)*	*Error (%)*	*Time (s)*	*Error (%)*	*Time (s)*
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

†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 italics. 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) central processor unit E5-2660 at 2.60 GHz.

boosting machines (implemented in the R package `gbm` (Ridgeway, 2017)), random forests (R package `randomForest` (Liaw and Wiener, 2002)), linear discriminant analysis (R package `MASS` (Venables and Ripley, 2002)) and deep neural net (R package `mxnet` (Chen *et al.*, 2015)). Out of 16 benchmark data sets, kernel DWD has the least prediction error on 10 examples, both random forests and deep neural net have the least on two cases, and both gradient boosting machines and linear discriminant analysis have the least on one data set.

6. Discussion

In the present paper we have considered the standard classification problem under 0–1-loss. In many applications we may face the so-called non-standard classification problem. For example,

Table 5. Mean misclassification rates and computation time for DWD with a data-driven q and four commonly used classifiers on 16 benchmark data sets†

Data	n	p	*Results for the following methods:*									
			$DWD_{\text{data-driven } q}$		*Gradient boosting*		*Random forests*		*Linear discriminant analysis*		*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)

†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 italics. The standard error of the mean is given in parentheses.

observed data may be collected via biased sampling and/or we need to consider unequal costs for different types of misclassification. Qiao *et al.* (2010) introduced 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 as

$$\min_{\beta_0,\boldsymbol{\beta}} \left\{ \sum_{i=1}^{n} w(y_i) \left(\frac{1}{r_i} + c\xi_i \right) \right\}, \qquad \text{subject to } r_i = y_i(\beta_0 + \mathbf{x}_i^{\mathrm{T}}\boldsymbol{\beta}) + \xi_i \geqslant 0 \text{ and } \boldsymbol{\beta}^{\mathrm{T}}\boldsymbol{\beta} = 1, \quad (6.1)$$

which can be further generalized to weighted kernel DWD:

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

Qiao *et al.* (2010) gave the expressions for $w(y_i)$ for various non-standard classification problems. Qiao *et al.* (2010) solved weighted DWD (6.1) with $q = 1$ based on SOCP. The MM procedure that is developed in this paper can easily accommodate the weight factors $w(y_i)$ to solve weighted DWD and weighted kernel DWD. We implemented the weighted DWD in the R package `kerndwd`.

Acknowledgements

We thank the Joint Editor, an Associate Editor and three referees for their helpful comments that greatly improved this work. We thank Professor Defeng Sun and Professor Kim-Chuan Toh for sharing the MATLAB toolbox `DWDLarge` that implements the inexact symmetric Gauss–Seidel alternating direction method of multipliers algorithm. Zou's research was partially supported by National Science Foundation grant DMS-1505111.

Appendix A: Technical proofs

A.1. Proof of lemma 1

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

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

$$G''(\eta_i) = \frac{q(q+1)}{(v_i+\eta_i)^{q+2}} > 0.$$

If

$$v_i > \left(\frac{q}{c}\right)^{1/(q+1)},$$

then $G'(\eta_i) > 0$ for all $\eta_i \geqslant 0$, and $\eta_i^* = 0$ is the minimizer. If

$$v_i \leqslant \left(\frac{q}{c}\right)^{1/(q+1)},$$

then

$$\eta_i^* = \left(\frac{q}{c}\right)^{1/(q+1)} - v_i$$

is the minimizer as $G'(\eta^*) = 0$ and $G''(\eta^*) > 0$.

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

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

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

We now simplify problem (6.2). Suppose that

$$t = \left(\frac{q}{q+1}\right)\left(\frac{q}{c}\right)^{-1/(q+1)}$$

and

$$t_1 = \left(\frac{1}{q+1}\right)\left(\frac{q}{c}\right)^{q/(q+1)}.$$

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

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

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

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

which can be further transformed to problem (3.1) with λ and t one-to-one correspondent.

A.2. Proof of lemma 2

We first prove expression (3.3). 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 $u > q/(q+1)$. Also $V_q'(u)$ is continuous on $[q/(q+1), \infty)$ and differentiable on $(q/(q+1), \infty)$.

If both u_1 and $u_2 > q/(q+1)$, then the mean value theorem implies that there exists $u^{**} > q/(q+1)$:

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

If $u_1 > q/(q+1)$ and $u_2 \leqslant q/(q+1)$, then

$$V_q'(u_2) = V_q'\left(\frac{q}{q+1}\right) = -1.$$

The mean value theorem implies that there exists $u^{**} > q/(q+1)$ satisfying

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

If both u_1 and $u_2 \leqslant q/(q+1)$, $V_q'(u_1) = V_q'(u_2) = -1$. 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 expressions (A.2)–(A.4), we prove expression (3.3).

We now prove expression (3.4). Let

$$\nu(a) \equiv \frac{(q+1)^2}{2q} a^2 - V_q(a).$$

From expression (3.3), it is not difficult to show that

$$\nu'(a) = \frac{(q+1)^2}{q} a - V_q'(a)$$

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 expression (3.4) directly.

A.3. Proof of lemma 3

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

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

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

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

We see that

(a) when $\eta>0.5$, $\partial\zeta(f)/\partial f=0$ only when

$$f=\tilde{f}\equiv\frac{q}{q+1}\left(\frac{\eta}{1-\eta}\right)^{1/(q+1)},$$

and

(b) when $\eta<0.5$, $\partial\zeta(f)/\partial f=0$ only when

$$f=\tilde{f}\equiv-\frac{q}{q+1}\left(\frac{1-\eta}{\eta}\right)^{1/(q+1)}.$$

Hence the convexity of the function ζ implies that $\tilde{f}$ is the minimizer of $\zeta(f)$.

A.4. Proof of lemma 4

As $\tilde{f}(\mathbf{x})$ was defined in equation (4.2), we see that, for each $\mathbf{x}$,

$$\begin{aligned}\zeta\{\tilde{f}(\mathbf{x})\}&\equiv\eta(\mathbf{x})V_q\{\tilde{f}(\mathbf{x})\}+\{1-\eta(\mathbf{x})\}V_q\{-\tilde{f}(\mathbf{x})\}\\&=\begin{cases}\eta(\mathbf{x})+\{1-\eta(\mathbf{x})\}^{1/(q+1)}\eta(\mathbf{x})^{q/(q+1)}, & \text{if } \eta(\mathbf{x})\leqslant\frac{1}{2},\\ 1-\eta(\mathbf{x})+\eta(\mathbf{x})^{1/(q+1)}\{1-\eta(\mathbf{x})\}^{q/(q+1)}, & \text{if } \eta(\mathbf{x})>\frac{1}{2},\end{cases}\\&=\tfrac{1}{2}\{1-|2\eta(\mathbf{x})-1|\}+\tfrac{1}{2}\{1+|2\eta(\mathbf{x})-1|\}^{1/(q+1)}\{1-|2\eta(\mathbf{x})-1|\}^{q/(q+1)}.\end{aligned}$$

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

$$\gamma(a)\equiv1-\frac{1}{2}(1-a)-\frac{1}{2}(1+a)^{1/(q+1)}(1-a)^{q/(q+1)}-\frac{q}{q+1}a,$$

$$\begin{aligned}\gamma'(a)&=\frac{1}{2}-\frac{1}{2(q+1)}\left(\frac{1-a}{1+a}\right)^{q/(q+1)}+\frac{q}{2(q+1)}\left(\frac{1+a}{1-a}\right)^{1/(q+1)}-\frac{q}{q+1}\\&=\frac{1}{2(q+1)}-\frac{1}{2(q+1)}\left(\frac{1-a}{1+a}\right)^{q/(q+1)}+\frac{q}{2(q+1)}+\frac{q}{2(q+1)}\left(\frac{1+a}{1-a}\right)^{1/(q+1)}-\frac{q}{q+1}\geqslant0.\end{aligned}$$

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

$$1-\zeta\{\tilde{f}(\mathbf{x})\}\geqslant\frac{q}{q+1}|2\eta(\mathbf{x})-1|.$$

By $R(f)=E_{\mathbf{X}Y}[Y\neq\operatorname{sgn}(f(\mathbf{X})]=E_{\{\mathbf{X}:f(\mathbf{X})\geqslant0\}}[1-\eta(\mathbf{X})]+E_{\{\mathbf{X}:f(\mathbf{X})\leqslant0\}}[\eta(\mathbf{X})]$, we obtain

$$\begin{aligned}R(\hat{f}_n)-R(f^*)&=E_{\{\mathbf{X}:\hat{f}_n(\mathbf{X})\geqslant0,\ f^*(\mathbf{X})<0\}}[1-2\eta(\mathbf{X})]+E_{\{\mathbf{X}:\hat{f}_n(\mathbf{X})\leqslant0,\ f^*(\mathbf{X})>0\}}[2\eta(\mathbf{X})-1]\\&\leqslant E_{\{\mathbf{X}:\hat{f}_n(\mathbf{X})f^*(\mathbf{X})\leqslant0\}}|2\eta(\mathbf{X})-1|\\&\leqslant\frac{q+1}{q}E_{\{\mathbf{X}:\hat{f}_n(\mathbf{X})f^*(\mathbf{X})\leqslant0\}}[1-\zeta\{\tilde{f}(\mathbf{X})\}].\end{aligned}\tag{A.5}$$

Since $f^*(\mathbf{X})$ and $\tilde{f}(\mathbf{X})$ share the same sign, $\hat{f}_n(\mathbf{X})f^*(\mathbf{X})\leqslant0$ implies that $\hat{f}_n(\mathbf{X})\tilde{f}(\mathbf{X})\leqslant0$. When $\hat{f}_n(\mathbf{X})\tilde{f}(\mathbf{X})\leqslant0$, 0 is between $\hat{f}_n(\mathbf{X})$ and $\tilde{f}(\mathbf{X})$, and thus the convexity of ζ indicates that $\zeta\{\tilde{f}(\mathbf{X})\}\leqslant\zeta(0)=1\leqslant\zeta\{\hat{f}_n(\mathbf{X})\}$. From inequality (A.5) we conclude that

$$
\begin{aligned}
R(\hat{f}_n) - R(f^*) &\leqslant \frac{q+1}{q} E_{\{\mathbf{X}:\hat{f}_n(\mathbf{X})f^*(\mathbf{X})\leqslant 0\}}[\zeta\{\hat{f}_n(\mathbf{X})\} - \zeta\{\tilde{f}(\mathbf{X})\}] \\
&\leqslant \frac{q+1}{q} E_{\mathbf{X}}[\zeta\{\hat{f}_n(\mathbf{X})\} - \zeta\{\tilde{f}(\mathbf{X})\}] \\
&= \frac{q+1}{q} E_{\mathbf{X}Y}[V_q\{Y\hat{f}_n(\mathbf{X})\} - V_q\{Y\tilde{f}(\mathbf{X})\}] \\
&= \frac{q+1}{q}(\varepsilon_{\mathrm{A}} + \varepsilon_{\mathrm{E}}).
\end{aligned}
$$

A.5. Proof of theorem 1

A.5.1. Part (a)

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

$$
|E_{\mathbf{X}Y}[V_q\{Yf_\epsilon(\mathbf{X})\}] - E_{\mathbf{X}Y}[V_q\{Y\tilde{f}(\mathbf{X})\}]| < \epsilon. \tag{A.6}
$$

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

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

We have that

$$
0 \leqslant E_{\mathbf{X}Y}[V_q\{Yf_\delta(\mathbf{X})\}] - E_{\mathbf{X}Y}[V_q\{Y\tilde{f}(\mathbf{X})\}] = \kappa_+ + \kappa_-,
$$

where

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

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

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

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

$$
0 \leqslant E_{\mathbf{X}Y}[V_q\{Yf_\delta(\mathbf{X})\}] - E_{\mathbf{X}Y}[V_q\{Y\tilde{f}(\mathbf{X})\}] \leqslant \kappa_+ + \kappa_- < \epsilon/3. \tag{A.7}
$$

By Lusin's theorem, there is a continuous function $\varrho(\mathbf{X})$ such that $P\{\varrho(\mathbf{X}) \neq f_\delta(\mathbf{X})\} \leqslant \epsilon(q+1)/(6q)$. Note that $\sup_{\mathbf{X}} |f_\delta(\mathbf{X})| \leqslant q/(q+1)$. Define

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

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

$$
\begin{aligned}
|E_{\mathbf{X}Y}[V_q\{Yf_\delta(\mathbf{X})\}] - E_{\mathbf{X}Y}[V_q\{Y\tau(\mathbf{X})\}]| &\leqslant E_{\mathbf{X}}|f_\delta(\mathbf{X}) - \tau(\mathbf{X})| \\
&= E_{\{\mathbf{X}:\tau(\mathbf{X})\neq f_\delta(\mathbf{X})\}}|f_\delta(\mathbf{X}) - \tau(\mathbf{X})| \\
&\leqslant \frac{2q}{q+1}\,\frac{\epsilon(q+1)}{6q} = \frac{\epsilon}{3},
\end{aligned} \tag{A8}
$$

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

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

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

$$
|E_{\mathbf{X}Y}[V_q\{Yf_\epsilon(\mathbf{X})\}] - E_{\mathbf{X}Y}[V_q\{Y\tau(\mathbf{X})\}]| < \sup_{\mathbf{X}} |f_\epsilon(\mathbf{X}) - \tau(\mathbf{X})| < \frac{\epsilon}{3}. \tag{A.9}
$$

By combining expressions (A.7)–(A.9) we obtain inequality (A.6).

A.5.2. Part (b)

In part (b) we bound the estimation error $\varepsilon_{\mathrm{E}}(\hat{f}_n)$. The proof of the estimation error approaching 0 essentially only requires the loss function to be Lipschitz continuous, which holds for the DWD case. Note that an RKHS has the following reproducing property (Wahba, 1990; Hastie *et al.*, 2009):

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

Fix any $\epsilon > 0$. By the Karush–Kuhn–Tucker condition of expression (4.3) and the representer theorem, we have

$$
\frac{1}{n}\sum_{i=1}^{n} V_q'\{y_i\hat{f}_n(\mathbf{x}_i)\}y_iK(\mathbf{x}_i, \mathbf{x}) + 2\lambda_n\hat{f}_n(\mathbf{x}) = 0. \tag{A.10}
$$

We define $\hat{f}^{[k]}$ as the solution of expression (4.3) when the kth datum is excluded from training data:

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

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

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

By the reproducing property, we further have

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

where the equality at the end holds by equation (A.10). Thus, by the Cauchy–Schwarz inequality,

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

which implies that

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

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

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

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

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

and therefore

$$\frac{1}{n}\sum_{k=1}^{n} V_q\{y_k\hat{f}^{[k]}(\mathbf{x}_k)\} \leqslant \frac{1}{n}\sum_{k=1}^{n} V_q\{y_k\hat{f}_n(\mathbf{x}_k)\} + \frac{B}{n\lambda_n}. \tag{A.11}$$

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

$$E_{\mathbf{X}Y}[V_q\{Yf^*_\epsilon(\mathbf{X})\}] \leqslant \inf_{f\in\mathcal{H}_K} E_{\mathbf{X}Y}[V_q\{Yf(\mathbf{X})\}] + \epsilon/3.$$

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

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

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

$$E_{\mathbf{T}_n}\left[\frac{1}{n}\sum_{k=1}^{n} V_q\{y_k\hat{f}^{[k]}(\mathbf{x}_k)\}\right] = \frac{1}{n}\sum_{k=1}^{n} E_{\mathbf{T}_n}[V_q\{y_k\hat{f}^{[k]}(\mathbf{x}_k)\}] = E_{\mathbf{T}_{n-1}}[E_{\mathbf{X}Y}V_q\{Y\hat{f}_{n-1}(\mathbf{X})\}]. \tag{A.12}$$

By combining expressions (A.11) and (A.12) we have

$$E_{\mathbf{T}_{n-1}}[E_{\mathbf{X}Y}[V_q\{Y\hat{f}_{n-1}(\mathbf{X})\}]] \leqslant \inf_{f\in\mathcal{H}_K} E_{\mathbf{X}Y}[V_q\{Yf(\mathbf{X})\}] + \lambda_n\|f^*_\epsilon\|^2_{\mathcal{H}_K} + \frac{B}{n\lambda_n} + \frac{\epsilon}{3}.$$

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

$$E_{\mathbf{T}_{n-1}}[E_{\mathbf{X}Y}[V_q\{Y\hat{f}_{n-1}(\mathbf{X})\}]] \leqslant \inf_{f\in\mathcal{H}_K} E_{\mathbf{X}Y}[V_q\{Yf(\mathbf{X})\}] + \epsilon.$$

Because ϵ is arbitrary and $E_{\mathbf{T}_{n-1}}[E_{\mathbf{X}Y}[V_q\{Y\hat{f}_{n-1}(\mathbf{X})\}]] \geqslant \inf_{f\in\mathcal{H}_K} E_{\mathbf{X}Y}[V_q\{Yf(\mathbf{X})\}]$, we have

$$\lim_{n\to\infty} E_{\mathbf{T}_{n-1}}[E_{\mathbf{X}Y}[V_q\{Y\hat{f}_{n-1}(\mathbf{X})\}]] = \inf_{f\in\mathcal{H}_K} E_{\mathbf{X}Y}[V_q\{Yf(\mathbf{X})\}],$$

which equivalently indicates that $\lim_{n\to\infty} E_{\mathbf{T}_n}[\varepsilon_{\mathrm{E}}(\hat{f}_n)] = 0$. Since $\varepsilon_{\mathrm{E}}(\hat{f}_n) \geqslant 0$, then by Markov inequality we prove part (b).

References

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

Alizadeh, F. and Goldfarb, D. (2004) Second-order cone programming. *Math. Programmng* B, **95**, 3–51.

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

Bartlett, P., Jordan, M. and McAuliffe, J. (2006) Convexity, classification, and risk bounds. *J. Am. Statist. Ass.*, **101**, 138–156.
Bartlett, P. and Shawe-Taylor, J. (1999) Generalization performance of support vector machines and other pattern classifiers. In *Advances in Kernel Methods–Support Vector Learning* (eds B. Schölkopf, C. J. Burges and A. J. Smola), vol. 4, pp. 43–54. Cambridge: MIT Press.
Boyd, S. and Vandenberghe, L. (2004) *Convex Optimization*. Cambridge: Cambridge University Press.
Breiman, L. (2001) Random forests. *Mach. Learn.*, **45**, 5–32.
Chen, T., Li, M., Li, Y., Lin, M., Wang, N., Wang, M., Xiao, T., Xu, B., Zhang, C. and Zhang, Z. (2015) Mxnet: a flexible and efficient machine learning library for heterogeneous distributed systems. *Preprint*. (Available from `https://arxiv.org/abs/1512.01274`.)
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. *Math. Programmng*, **161**, 237–270.
De Leeuw, J. and Heiser, W. (1977) Convergence of correction matrix algorithms for multidimensional scaling. In *Geometric Representations of Relational Data* (ed. J. C. Lingoes), pp. 735–752. Ann Arbor: Mathesis.
Fernández-Delgado, M., Cernadas, E., Barro, S. and Amorim, D. (2014) Do we need hundreds of classifiers to solve real world classification problems? *J. Mach. Learn. Res.*, **15**, 3133–3181.
Freund, Y. and Schapire, R. (1997) A decision-theoretic generalization of on-line learning and an application to boosting. *J. Comput. Syst. Sci.*, **55**, 119–139.
Hall, P., Marron, J. S. and Neeman, A. (2005) Geometric representation of high dimension, low sample size data. *J. R. Statist. Soc.* B, **67**, 427–444.
Hastie, T., Tibshirani, R. and Friedman, J. (2009) *The Elements of Statistical Learning: Prediction, Inference, and Data Mining*, 2nd edn. Berlin: Springer.
Hsieh, C., Si, S. and Dhillon, I. (2014) A divide-and-conquer solver for kernel support vector machines. In *Proc. Int. Conf. Machine Learning*, pp. 566–574.
Huang, H., Liu, Y., Du. Y., Perou, C., Hayes, D., Todd, M. and Marron, J. S. (2013) Multiclass distance-weighted discrimination. *J. Computnl Graph. Statist.*, **22**, 953–969.
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**, 1182–1183.
Hunter, D. and Lange, K. (2004) A tutorial on MM algorithms. *Am. Statistn*, **58**, 30–37.
Hunter, D. and Li, R. (2005) Variable selection using MM algorithms. *Ann. Statist.*, **33**, 1617–1642.
Jaakkola, T. and Haussler, D. (1999) Probabilistic kernel regression models. In *Proc. 7th Int. Wrkshp Artificial Intelligence and Statistics*.
Karatzoglou, A., Smola, A., Hornik, K. and Zeileis, A. (2004) kernlab—an S4 package for kernel methods in R. *J. Statist. Softwr.*, **11**, 1–20.
Lam, X., Marron, J. S., Sun, D. and Toh, K. C. (2017) Fast algorithms for large scale generalized distance weighted discrimination. *Preprint*. (Available from `https://arxiv.org/abs/1604.05473`.)
Lange, K., Hunter, D. and Yang, I. (2000) Optimization transfer using surrogate objective functions. *J. Computnl Graph. Statist.*, **9**, 1–20.
Lange, K. and Zhou, H. (2014) MM algorithms for geometric and signomial programming. *Math. Programmng*, **143**, 339–356.
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. *Math. Programmng*, **155**, 333–373.
Liaw, A. and Wiener, M. (2002) Classification and regression by randomForest. *R News*, **2**, no. 3, 18–22.
Lichman, M. (2013) UCI Machine Learning Repository. School of Information and Computer Science, University of California at Irvine, Irvine. (Available from `http://archive.ics.uci.edu/ml`.)
Lin, Y. (2002) Support vector machines and the Bayes rule in classification. *Data Minng Knowl. Discov.*, **6**, 259–275.
Lin, Y. (2004) A note on margin-based loss functions in classification. *Statist. Probab. Lett.*, **68**, 73–82.
Lin, Y., Lee, Y. and Wahba, G. (2002) Support vector machines for classification in nonstandard situations. *Mach. Learn.*, **46**, 191–202.
Liu, Y., Zhang, H. and Wu, Y. (2011) Hard or soft classification?: Large-margin unified machines. *J. Am. Statist. Ass.*, **106**, 166–177.
Marron, J. S. (2013) Smoothing, functional data analysis, and distance weighted discrimination software. (Available from `http://www.unc.edu/~marron/marron_software.html`.)
Marron, J. S. (2015) Distance-weighted discrimination. *Wiley Interdisc. Rev. Computnl Statist.*, **7**, 109–114.
Marron, J. S., Todd, M. and Ahn, J. (2007) Distance weighted discrimination. *J. Am. Statist. Ass.*, **102**, 1267–1271.
Meyer, D., Dimitriadou, E., Hornik, K., Weingessel, A. and Leisch, F. (2015) *R Package e1071*. (Available from `https://cran.r-project.org/web/packages/e1071/index.html`.)
Micchelli, C., Xu, Y. and Zhang, H. (2006). Universal kernels. *J. Mach. Learn. Res.*, **7**, 2651–2667.
Platt, J. (1999) Fast training of support vector machines using sequential minimal optimization. In *Advances in Kernel Methods—Support Vector Learning*, vol. 12 (eds B. Schölkopf, C. J. Burges and A. J. Smola), pp. 185–208. Cambridge: MIT Press.
Qiao, X. and Zhang, L. (2015a) Distance-weighted support vector machine. *Statist. Interfc.*, **8**, 331–345.
Qiao, X. and Zhang, L. (2015b) Flexible high-dimensional classification machines and their asymptotic properties. *J. Mach. Learn. Res.*, **16**, 1547–1572.

Qiao, X., Zhang, H., Liu, Y., Todd, M. and Marron, J. S. (2010) Weighted distance weighted discrimination and its asymptotic properties. *J. Am. Statist. Ass.*, **105**, 401–414.
Ridgeway, G. (2017) gbm: generalized boosted regression models. *R Package 2.9.0*. (Available from `https://cran.r-project.org/web/packages/gbm/gbm.pdf`.)
Shawe-Taylor, J. and Cristianini, N. (2000) Margin distribution and soft margin. In *Advances in Kernel Methods—Support Vector Learning*, vol. 19 (eds A. J. Smola, P. Bartlett, B. Schölkopf and D. Schuurmans), pp. 349–358. Cambridge: MIT Press.
Steinwart, I. (2001) On the influence of the kernel on the consistency of support vector machines. *J. Mach. Learn. Res.*, **2**, 67–93.
Vapnik, V. (1995) *The Nature of Statistical Learning Theory*. Berlin: Springer.
Vapnik, V. (1998) *Statistical Learning Theory*. Chichester: Wiley.
Venables, W. and Ripley, B. (2002) *Modern Applied Statistics with S*, 4th edn. Berlin: Springer.
Wahba, G. (1990) *Spline Models for Observational Data*. Philadelphia: Society for Industrial and Applied Mathematics.
Wahba, G. (1999) Support vector machines, reproducing kernel Hilbert spaces and the randomized GACV. In *Advances in Kernel Methods–Support Vector Learning*, vol. 6 (eds B. Schölkopf, C. J. Burges and A. J. Smola), pp. 69–87. Cambridge: MIT press.
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 Proc.*, vol. 20, pp. 313–331. Reading: Addison-Wesley.
Wang, B. and Zou, H. (2016) Sparse distance weighted discrimination. *J. Computnl Graph. Statist.*, **25**, 826–838.
Wu, T. and Lange, K. (2010) The MM alternative to EM. *Statist. Sci.*, **25**, 492–505.
Yang, Y. and Zou, H. (2013) An efficient algorithm for computing the HHSVM and its generalizations. *J. Computnl Graph. Statist.*, **22**, 396–415.
Zhang, T. (2004) Statistical behavior and consistency of classification methods based on convex risk minimization. *Ann. Statist.*, **32**, 56–134.
Zhang, Y., Duchi, J. and Wainwright, M. (2015) Divide and conquer kernel ridge regression: a distributed algorithm with minimax optimal rates. *J. Mach. Learn. Res.*, **16**, 3299–3340.
Zhou, H. and Lange, K. (2010) MM algorithms for some discrete multivariate distributions. *J. Computnl Graph. Statist.*, **19**, 645–665.
Zhu, J. and Hastie, T. (2005) Kernel logistic regression and the import vector machine. *J. Computnl Graph. Statist.*, **14**, 185–205.
Zou, H. and Li, R. (2008) One-step sparse estimates in nonconcave penalized likelihood models. *Ann. Statist.*, **36**, 1509–1533.