Math. Program., Ser. B (2018) 169:277–305
https://doi.org/10.1007/s10107-017-1209-5

FULL LENGTH PAPER

Robust multicategory support vector machines using difference convex algorithm

Chong Zhang[1] · Minh Pham[2,3] · Sheng Fu[4] · Yufeng Liu[5]

Received: 4 February 2016 / Accepted: 9 November 2017 / Published online: 29 November 2017

Abstract The support vector machine (SVM) is one of the most popular classification methods in the machine learning literature. Binary SVM methods have been extensively studied, and have achieved many successes in various disciplines. However, generalization to multicategory SVM (MSVM) methods can be very challenging. Many existing methods estimate k functions for k classes with an explicit sum-to-zero constraint. It was shown recently that such a formulation can be suboptimal. Moreover, many existing MSVMs are not Fisher consistent, or do not take into account the effect of outliers. In this paper, we focus on classification in the angle-based framework, which is free of the explicit sum-to-zero constraint, hence more efficient, and propose two robust MSVM methods using truncated hinge loss functions. We show that our new classifiers can enjoy Fisher consistency, and simultaneously alleviate the impact of outliers to achieve more stable classification performance. To implement our proposed classifiers, we employ the difference convex algorithm for efficient computation. Theoretical and numerical results obtained indicate that for problems with potential outliers, our robust angle-based MSVMs can be very competitive among existing methods.

✉ Yufeng Liu
yfliu@email.unc.edu

1 Department of Statistics and Actuarial Science, University of Waterloo, Waterloo, ON N2L 3G1, Canada

2 Statistical and Applied Mathematical Sciences Institute (SAMSI), Durham, NC, USA

3 Department of Statistics, University of Virginia, Charlottesville, VA, USA

4 University of Chinese Academy of Sciences, Beijing, China

5 Department of Statistics and Operations Research, Department of Genetics, Department of Biostatistics, Carolina Center for Genome Sciences, Lineberger Comprehensive Cancer Center, University of North Carolina at Chapel Hill, Chapel Hill, NC, USA

Keywords Difference convex algorithm · Fisher consistency · Outlier · Truncated hinge loss

Mathematics Subject Classification 90C26 · 62H30

1 Introduction

Classification is an important type of supervised learning problems in machine learning. Given a training dataset with both inputs and class labels available for all subjects, one important goal of classification is to build a classification model, also known as a classifier, to predict the class label accurately for new subjects with inputs only. There are many existing classifiers in the literature [14]. Among various classifiers, the Support Vector Machine (SVM, [6,8]) is a popular method that was first introduced in the machine learning literature. As a typical margin-based classifier, the SVM has achieved many successes in various scientific disciplines, such as artificial intelligence, cancer research, and econometrics. Theoretical properties of standard SVMs have been well established, including the Fisher consistency (more details are given in Sect. 2.2; see also, [29,30]), and generalization error bounds [5,40]. For a comprehensive introduction of SVMs, we refer the readers to [10,14], among others.

Binary SVM methods have been extensively studied in the literature. In particular, the original binary SVM searches for a hyperplane in the feature space that can maximally separate the two classes, and uses a single classification function for prediction. The signed distance between an observation and the separating boundary is called the functional margin. One can verify that the corresponding optimization problem is equivalent to using the hinge loss function on functional margins of training observations. Many useful results for binary SVMs, including the feature selection properties for high dimensional learning [24,45], have been obtained. Empirical studies in the literature have confirmed the usefulness of binary SVMs (see, for example, [1,7,16]).

Despite the success, how to extend binary SVM methods to address multicategory problems is a challenging problem. In the simultaneous margin-based classification framework, a common approach to handle problems with k different labels is to use k classification functions, and the corresponding prediction rule is based on which function is the largest. To reduce the parameter space and to ensure theoretical properties of the classifier, a sum-to-zero constraint is often imposed on these k functions. In the literature, many existing Multicategory SVM (MSVM) methods follow this procedure, including [9,13,26,31,32,43]. Recently, [49] suggested that this procedure can be inefficient and suboptimal, and proposed angle-based classification as a new margin-based classification framework. Zhang and Liu [49] showed that angle-based classifiers can enjoy better prediction accuracies and faster computational speeds, hence are very competitive.

For the extension from binary SVM to MSVM methods in the angle-based framework, there are still many open problems. For example, [49] showed that the naive SVM generalization is not Fisher consistent. To overcome this drawback, [49] proposed to use a large-margin unified machine loss function with appropriate parameters to approximate the hinge loss for consistency (more details of the large-margin unified machine can be found in [33]). Moreover, [50] proposed a new reinforced MSVM classifier in the angle-based framework that can enjoy Fisher consistency. Another open problem is that existing angle-based MSVM methods may not be robust against outliers. In particular, consider the feature space of the predictors, and assume that a training observation with label 2 is deeply buried in the group of observations with label 1. In this case, the fitted functional margin of this outlier, whether using the classifier in [49] or [50], would be negative with a large absolute value. This leads to a very large loss value for this single observation. Consequently, the fitted angle-based MSVM classifier can be unstable and suboptimal. A similar phenomenon was also observed by [48]. In particular, [48] proposed to employ truncated hinge loss functions for MSVMs using k classification functions and the sum-to-zero constraint. Because the angle-based framework can enjoy better efficiency and accuracy, it is desirable to explore how to develop angle-based MSVM classifiers that can achieve both Fisher consistency and robustness against potential outliers simultaneously. To fill this gap, in this paper, we propose two MSVM methods in the angle-based framework using truncated hinge loss functions. We show that both classifiers can enjoy Fisher consistency and other attractive theoretical properties. Our numerical results demonstrate that the new robust MSVM classifiers are very competitive among existing methods.

Because the truncated hinge loss functions we employ in this paper are non-convex, the corresponding optimization problems are more involved than the original quadratic programming or linear programming for MSVM methods. To overcome this difficulty, we notice that the nonconvex objective function can have a difference of convex functions decomposition (DC), and propose to apply the difference of convex functions algorithm, i.e., difference convex algorithm (DCA) [20]. The DCA decomposes the original non-convex problems into a sequence of convex subproblems, and each subproblem can be solved efficiently. We propose several robust SVM formulations combining truncated hinge loss functions and a variety of regularizers. We further apply efficient algorithms to solve the corresponding convex subproblems. Through numerical examples, we demonstrate that the DCA can work very well for our new classifiers.

The rest of this article is organized as follows. In Sect. 2, we first give a review of some existing MSVM classifiers, then introduce our robust angle-based MSVM methods. In Sect. 3, we show how to implement the DCA to solve the corresponding optimization problems, and provide convergence results to our solution methods. Some statistical learning theory, including Fisher consistency of our new methods, is obtained in Sect. 4. We perform numerical studies to demonstrate the effectiveness of our new methods in Sect. 5. All proofs are collected in the “Appendix”.

2 Methodology

In this section, we first give a brief review of some margin-based classification methods in Sect. 2.1, then propose our new classifiers in Sect. 2.2.

2.1 Review of some margin-based classifiers

Denote by $P(\boldsymbol{X}, Y)$ the underlying joint distribution of $(\boldsymbol{X}, Y)$, where $\boldsymbol{X}$ is the vector of predictors, and Y is the label. The learning goal is to find a classifier with the prediction rule $\hat{y}(\cdot)$, such that for any future observations, the misclassification rate $E[I\{Y \neq \hat{y}(\boldsymbol{X})\}]$ is minimized. Here I is the indicator function, and the expectation is taken with respect to the distribution P. Given a set of training observations $\{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_n, y_n)\}$ from P, an intuitive approach is to find a classifier by minimizing the empirical prediction error $\frac{1}{n}\sum_{i=1}^{n} I\{y_i \neq \hat{y}(\boldsymbol{x}_i)\}$. However, because the indicator function is discontinuous, such an optimization can be very difficult.

To circumvent this difficulty, it is common to use a surrogate loss function in place of the indicator function. For binary classifiers with $Y \in \{+1, -1\}$, one uses a single function $f(\cdot)$ for classification, and the prediction rule is $\hat{y}(\boldsymbol{x}) = \text{sign}\{f(\boldsymbol{x})\}$. In this case, the indicator function $I\{y \neq \hat{y}(\boldsymbol{x})\}$ is equivalent to $I\{yf(\boldsymbol{x}) < 0\}$, and the term $yf(\boldsymbol{x})$ is referred to as the functional margin. Binary margin-based classifiers use surrogate loss functions to encourage large functional margins. In particular, the corresponding optimization problem is typically

$$\min_{f \in \mathscr{F}} \frac{1}{n} \sum_{i=1}^{n} \ell\{yf(\boldsymbol{x})\} + \lambda J(f),$$

where $\mathscr{F}$ is the functional space, $\ell(\cdot)$ is the surrogate loss function, $J(f)$ is a penalty to prevent overfitting, and λ is a tuning parameter to balance the loss and penalty terms. Different binary margin-based classifiers use different surrogate loss functions. For example, the standard SVM uses the hinge loss $\ell(u) = [1-u]_+$, where $[u]_+ = \max(u, 0)$, logistic regression [27] uses the deviance loss $\ell(u) = \log\{1 + \exp(-u)\}$, and AdaBoost in boosting [12] is shown to be approximately equivalent to using the exponential loss $\ell(u) = \exp(-u)$.

For multicategory problems, how to define the prediction rule and functional margins becomes more involved. The details of angle-based classification are as follows. For a problem with k classes, consider a centered simplex with k vertices $\boldsymbol{W} = \{\boldsymbol{W}_1, \ldots, \boldsymbol{W}_k\}$ in $\mathbb{R}^{k-1}$ with

$$\boldsymbol{W}_j = \begin{cases} (k-1)^{-1/2}\mathbf{1} & j = 1, \\ -(1+k^{1/2})/\{(k-1)^{3/2}\}\mathbf{1} + \{k/(k-1)\}^{1/2} e_{j-1} & 2 \leq j \leq k, \end{cases}$$

where $\mathbf{1}$ is a vector of 1, and e_j is a vector with its jth element 1 and 0 elsewhere. One can verify that the matrix $\boldsymbol{W}$ introduces a symmetric simplex in $\mathbb{R}^{k-1}$. Without loss of generality, assume that class j is assigned to $\boldsymbol{W}_j$. The angle-based classifiers map $\boldsymbol{x}$ into $\mathbb{R}^{k-1}$ using $k-1$ functions $\boldsymbol{f} = (f_1, \ldots, f_{k-1})$. This classification function vector $\boldsymbol{f}(\boldsymbol{x})$ defines k angles with respect to $\{\boldsymbol{W}_1, \ldots, \boldsymbol{W}_k\}$, namely, $\angle(\boldsymbol{W}_j, \boldsymbol{f})$; $j = 1, \ldots, k$. The prediction rule is based on which angle is the smallest, i.e., $\hat{y}(\boldsymbol{x}) = \text{argmin}_{j \in \{1,\ldots,k\}} \angle(\boldsymbol{W}_j, \boldsymbol{f})$. Note that the smaller $\angle(\boldsymbol{W}_j, \boldsymbol{f})$ is, the larger the corresponding inner product $\langle \boldsymbol{W}_j, \boldsymbol{f} \rangle$ would be. Thus, it is equivalent to maximizing $\langle \boldsymbol{W}_y, \boldsymbol{f}(\boldsymbol{x}) \rangle$ in the optimization, where $\langle \boldsymbol{W}_j, \boldsymbol{f}(\boldsymbol{x}) \rangle$ can be regarded as functional margins for angle-based methods. Zhang and Liu [49] proposed to use the following optimization problem for angle-based classification

$$\min_{f \in \mathscr{F}} \frac{1}{n} \sum_{i=1}^{n} \ell\{\langle \boldsymbol{W}_{y_i}, \boldsymbol{f}(\boldsymbol{x}_i)\rangle\} + \lambda J(\boldsymbol{f}), \tag{1}$$

where ℓ is a binary margin-based loss function. For angle-based classifiers, we note that the functional margins sum to zero implicitly. Zhang and Liu [49] showed that, without an explicit sum-to-zero constraint as that used by other existing methods, the angle-based classifier can achieve a faster computational speed, and better classification performance.

2.2 Robust angle-based support vector machines

A direct generalization of the SVM method in the angle-based framework is to use $\ell(u) = [1 - u]_+$ in (1). However, [49] proved that this naive MSVM is not Fisher consistent, and proposed to use a loss function in the large-margin unified machine family to approximate the hinge loss to achieve Fisher consistency. In particular, Fisher consistency is defined as follows. Consider an observation with fixed $\boldsymbol{X} = \boldsymbol{x}$, and denote by $P_j(\boldsymbol{x}) = \text{pr}(Y = j \mid \boldsymbol{X} = \boldsymbol{x})$ the class conditional probability of class $j \in \{1, \ldots, k\}$. One can verify that the best prediction rule, namely, the Bayes rule, which minimizes the misclassification rate, is $\hat{y}_{\text{Bayes}}(\boldsymbol{x}) = \text{argmax}_j \, P_j(\boldsymbol{x})$. For a classifier, denote by $\phi\{\boldsymbol{f}(\boldsymbol{x}), y\}$ its surrogate loss function for classification using $\boldsymbol{f}$ as the classification function, and $\hat{y}_{\boldsymbol{f}}$ the corresponding prediction rule. Define the conditional loss $S(\boldsymbol{x}) = E[\phi\{\boldsymbol{f}(\boldsymbol{X}), Y\} \mid \boldsymbol{X} = \boldsymbol{x}]$, where the expectation is taken with respect to the marginal distribution of $Y \mid \boldsymbol{X} = \boldsymbol{x}$. We call $\boldsymbol{f}^*(\boldsymbol{x}) = \text{arginf}_{\boldsymbol{f}} \, S(\boldsymbol{x})$ the theoretical minimizer of the conditional loss. Fisher consistency requires that $\hat{y}_{\boldsymbol{f}^*}(\boldsymbol{x}) = \hat{y}_{\text{Bayes}}(\boldsymbol{x})$. In other words, Fisher consistency means that if we are using infinitely many training observations and an appropriate functional space $\mathscr{F}$, then the obtained classifier can achieve the best prediction performance, which is a fundamental requirement for a classification method. Zhang et al. [50] proposed the following reinforced MSVM loss function in the angle-based framework,

$$\phi\{\boldsymbol{f}(\boldsymbol{x}), y\} = \gamma[(k-1) - \langle \boldsymbol{f}(\boldsymbol{x}), \boldsymbol{W}_y\rangle]_+ + (1-\gamma)\sum_{j \neq y}[1 + \langle \boldsymbol{f}(\boldsymbol{x}), \boldsymbol{W}_j\rangle]_+, \tag{2}$$

where $\gamma \in [0, 1]$ is the convex combination parameter. Note that the first term of (2) explicitly encourages $\langle \boldsymbol{f}(\boldsymbol{x}), \boldsymbol{W}_y\rangle$ to be large. The second term of (2) encourages $\langle \boldsymbol{f}(\boldsymbol{x}), \boldsymbol{W}_j\rangle$ to be small for $j \neq y$. This implicitly encourages $\langle \boldsymbol{f}(\boldsymbol{x}), \boldsymbol{W}_y\rangle$ to be large, because $\sum_{j=1}^{k}\langle \boldsymbol{f}(\boldsymbol{x}), \boldsymbol{W}_j\rangle = 0$. [50] showed that their reinforced MSVM method using this convex combination of hinge loss functions with $\gamma \in [0, 0.5]$ enjoys Fisher consistency.

Despite the progress, neither [49] nor [50] addressed the problem of potential outliers in practical problems. Consequently, the fitted classifiers may be suboptimal. In Fig. 1, we demonstrate the effect of outliers on the performance of the angle-based MSVM proposed by [49], using a simulated example. In particular, we first generalize a training dataset that has no outliers, and find the best angle-based MSVM classifier. We plot the corresponding classification boundaries on the left panel of Fig. 1. Next,

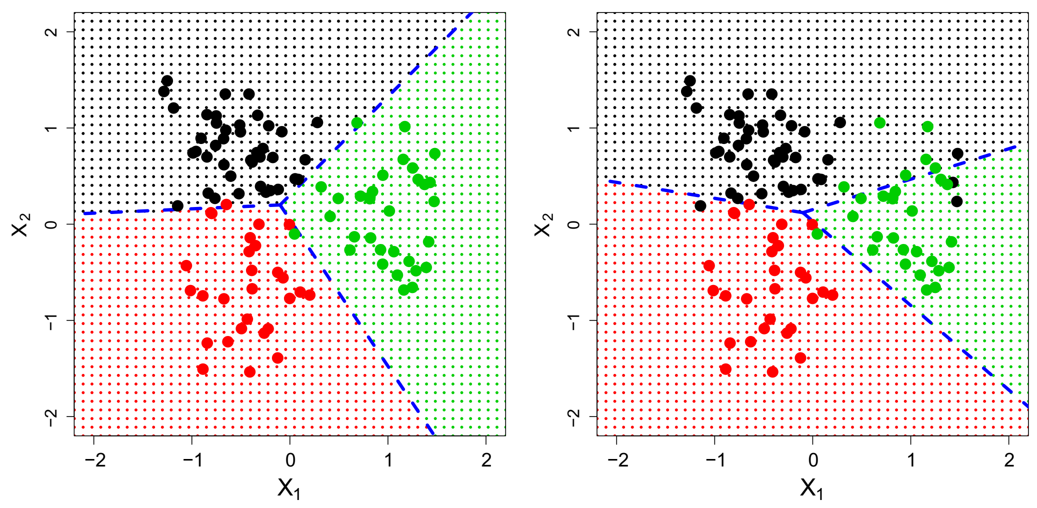

Fig. 1 The training dataset on the left panel has no potential outliers, whereas the training set on the right panel has three (black) potential outliers in the green group of observations. The blue dashed lines are the fitted classification boundaries (color figure online)

the class labels of some observations in one group are changed to another randomly. In practical problems, this type of unobserved mislabeling can be common, such as recording errors, a misdiagnosis of a patient in clinical trials. We then train a new angle-based MSVM classifier, and plot the corresponding classification boundaries on the right panel of Fig. 1. For both classifiers, we use L_2 penalized linear learning, and the best tuning parameters are selected via 5-fold cross validation. One can see that for the data without outliers, the angle-based method works well. In contrast, for the second classifier, the existence of outliers has a significant effect on the classification boundary estimation, and the prediction accuracy deteriorates.

To better understand the effect of outliers on classification performance, we consider the reinforced MSVM loss (2) as an illustrating example. The first loss function term, $[k-1-u]_+$, increases linearly when $u < k-1$ decreases, and the second loss term $[1+u]_+$ increases linearly when $u > -1$ increases. For a potential outlier such as those in Fig. 1, it is often to have $\langle \boldsymbol{f}(\boldsymbol{x}), \boldsymbol{W}_y \rangle$ being negative with a large absolute value. Moreover, because of the implicit sum-to-zero property $\sum_{j=1}^{k} \langle \boldsymbol{f}(\boldsymbol{x}), \boldsymbol{W}_j \rangle = 0$, some of $\langle \boldsymbol{f}(\boldsymbol{x}), \boldsymbol{W}_j \rangle$, $j \neq y$ can be positive with a large absolute value. This can result in a large $\phi\{\boldsymbol{f}(\boldsymbol{x}), y\}$ for this single observation, which, consequently, has a great influence on the final solution. For the angle-based MSVM using the approximate hinge loss, one can verify that a similar issue exists. Therefore, it is desirable to decrease the loss $\phi\{\boldsymbol{f}(\boldsymbol{x}), y\}$ for such outliers in MSVM methods. To this end, we propose to use truncated hinge loss functions in the angle-based classification framework. In particular, in this paper, we propose two such MSVM methods. We first explore how to implement the truncated hinge loss function for the reinforced MSVM by [50]. Then we propose a novel MSVM method in the angle-based framework, and show how to employ the truncated hinge loss function for this new classifier to alleviate the effect of potential outliers.

To begin with, define $H_s(u) = [s-u]_+$, and $G_s(u) = [s+u]_+$. One can rewrite (2) as

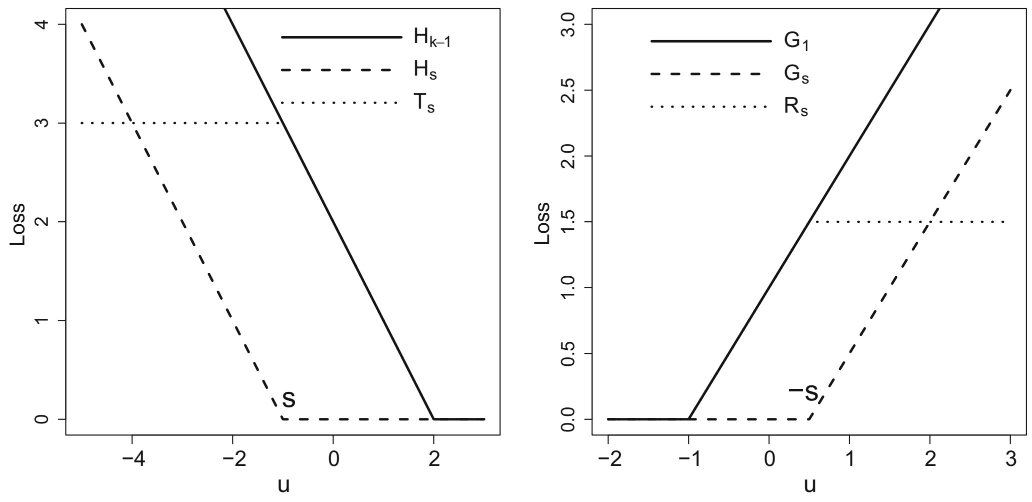

Fig. 2 Plots of the loss functions T_s (left) and R_s (right)

$$\phi\{\boldsymbol{f}(\boldsymbol{x}), y\} = \gamma H_{k-1}\{\langle \boldsymbol{f}(\boldsymbol{x}), \boldsymbol{W}_y\rangle\} + (1-\gamma)\sum_{j\neq y} G_1\{\langle \boldsymbol{f}(\boldsymbol{x}), \boldsymbol{W}_j\rangle\}.$$

To reduce the effect of outliers, we consider the truncated hinge losses $T_s(u) = H_{k-1}(u) - H_s(u)$ and $R_s(u) = G_1(u) - G_s(u)$. Here s is a parameter that controls the location of truncation. In this paper we assume that k is a fixed constant, thus in the notation of T_s we suppress the dependency on k. When $s > 0$, $T_s(u)$ and $R_s(u)$ are constant within $[-s, s]$. In this case, one can verify that the loss for some correctly classified observations is the same as that of those misclassified ones, which is not desirable. Hence, we set $s \leq 0$. For real applications, one can perform a data adaptive tuning method to select the best $s \leq 0$. In our numerical experience, the choice of $s = -1/(k-1)$ works well. We plot $T_s(u)$ and $R_s(u)$ in Fig. 2.

With $T_s(u)$ and $R_s(u)$ defined as above, we propose the following loss function for our first robust angle-based MSVM

$$\phi_1\{\boldsymbol{f}(\boldsymbol{x}), y\} = \gamma T_{(k-1)s}\{\langle \boldsymbol{f}(\boldsymbol{x}), \boldsymbol{W}_y\rangle\} + (1-\gamma)\sum_{j\neq y} R_s\{\langle \boldsymbol{f}(\boldsymbol{x}), \boldsymbol{W}_j\rangle\}, \tag{3}$$

where k is the number of classes. Note that $T_{(k-1)s} = H_{k-1}(u) - H_{(k-1)s}(u)$, and we choose $T_{(k-1)s}$ such that the locations of truncation in (3) sum to zero, which is consistent with the implicit sum-to-zero property $\sum_{j=1}^{k}\langle \boldsymbol{f}, \boldsymbol{W}_j\rangle = 0$ of angle-based classifiers. One can see that for any potential outlier $(\boldsymbol{x}, y)$ with a large $\langle \boldsymbol{f}(\boldsymbol{x}), \boldsymbol{W}_y\rangle < 0$, its loss $\phi_1\{\boldsymbol{f}(\boldsymbol{x}), y\}$ is upper bounded by a constant for any $\boldsymbol{f}$. Thus, the impact of outliers can be alleviated by our new method using the loss in (3).

Our second robust MSVM loss function is motivated by the following observation. An instance $(\boldsymbol{x}, y)$ is correctly classified using $\boldsymbol{f}$ if and only if $\langle \boldsymbol{f}(\boldsymbol{x}), \boldsymbol{W}_y\rangle > \langle \boldsymbol{f}(\boldsymbol{x}), \boldsymbol{W}_j\rangle$ for all $j \neq y$. This is equivalent to $\langle \boldsymbol{f}(\boldsymbol{x}), \boldsymbol{W}_y - \boldsymbol{W}_j\rangle > 0$ for all $j \neq y$, or $\min_{j\neq y}\{\langle \boldsymbol{f}(\boldsymbol{x}), \boldsymbol{W}_y - \boldsymbol{W}_j\rangle\} > 0$. Therefore, the theoretical misclassification error rate can be written as

$$E(I[\min_{j \neq Y}\{\langle \boldsymbol{f}(\boldsymbol{X}), \boldsymbol{W}_Y - \boldsymbol{W}_j \rangle\} \leq 0]). \tag{4}$$

To employ the SVM method in this framework, one can use a surrogate hinge loss function in place of the indicator function in (4)

$$\phi\{\boldsymbol{f}(\boldsymbol{x}), y\} = H_1[\min_{j \neq y}\{\langle \boldsymbol{f}(\boldsymbol{x}), (\boldsymbol{W}_y - \boldsymbol{W}_j)\rangle\}]. \tag{5}$$

However, for outliers with $\langle \boldsymbol{f}(\boldsymbol{x}), \boldsymbol{W}_y \rangle < 0$ and $|\langle \boldsymbol{f}(\boldsymbol{x}), \boldsymbol{W}_y \rangle| \gg 1$, we have the corresponding $\min_{j \neq y}\{\langle \boldsymbol{f}(\boldsymbol{x}), (\boldsymbol{W}_y - \boldsymbol{W}_j)\rangle\}$ being negative with a large absolute value. To reduce the loss for such observations, we propose to use the following loss function for our second robust angle-based MSVM,

$$\phi_2\{\boldsymbol{f}(\boldsymbol{x}), y\} = T_s[\min_{j \neq y}\{\langle \boldsymbol{f}(\boldsymbol{x}), (\boldsymbol{W}_y - \boldsymbol{W}_j)\rangle\}], \tag{6}$$

where $T_s = H_1(u) - H_s(u)$ with $s \leq 0$. One can verify that for any instance with very small $\langle \boldsymbol{f}(\boldsymbol{x}), \boldsymbol{W}_y \rangle$, its loss $\phi_2\{\boldsymbol{f}(\boldsymbol{x}), y\}$ is upper bounded by $1-s$. Therefore, compared to (5), the influence of outliers can be alleviated in our new robust angle-based SVM.

Note that for regular MSVM methods that use k functions for a k-class problem, [9,31,48] considered pairwise comparisons of the k functions for classification. To our knowledge, in the angle-based classification literature, there is no existing work that has considered pairwise comparisons of functional margins such as in (4), or the corresponding surrogate loss functions for the indicator function in (6). Hence, our second robust angle-based MSVM method is different from the methods mentioned above. We will examine the performance of these two robust methods in Sect. 5.

In the next section, we discuss how to implement the proposed classifiers using DCA.

3 Computational implementation using difference convex algorithm

In this section, we discuss how to implement our robust angle-based MSVM methods using DCA. DCA was introduced by Pham Dinh Tao in 1985 and further developed by Le Thi Hoai An and Pham Dinh Tao. More details about DCA and some of its recent developments can be found in [21–23,25]. We provide detailed algorithms for the two proposed MSVMs in Sects. 3.1 and 3.2. In this section, we focus on linear learning with L_2, L_1, and mixed $L_1 + L_2$ penalties.

3.1 Algorithm for robust angle-based MSVM 1

We consider the DC formulation of (3)

$$\min_{\boldsymbol{\beta}} G(\boldsymbol{\beta}) - H(\boldsymbol{\beta}),$$

where

$$G(\boldsymbol{\beta}) = \sum_{i=1}^{n} \left[\gamma H_{k-1}\{\langle \boldsymbol{f}(\boldsymbol{x}_i), \boldsymbol{W}_{y_i} \rangle\} + (1-\gamma) \sum_{j \neq y_i} G_1\{\langle \boldsymbol{f}(\boldsymbol{x}_i), \boldsymbol{W}_j \rangle\} \right] + \lambda J(\boldsymbol{f}),$$

and

$$H(\boldsymbol{\beta}) = \sum_{i=1}^{n} \left[\gamma H_s\{\langle \boldsymbol{f}(\boldsymbol{x}_i), \boldsymbol{W}_{y_i} \rangle\ \} + (1-\gamma) \sum_{j \neq y_i} G_s\{\langle \boldsymbol{f}(\boldsymbol{x}_i), \boldsymbol{W}_j \rangle\} \right].$$

For linear learning, we assume $f_q(\boldsymbol{x}) = \boldsymbol{x}^{\mathrm{T}}\boldsymbol{\beta}_q$, where $\boldsymbol{\beta}_q \in \mathbb{R}^p$ $(q = 1, \ldots, k-1)$ are the vectors of parameters that we are interested in. For brevity, we denote $\boldsymbol{f}(\boldsymbol{x})$ by $\boldsymbol{B}\boldsymbol{x}$, where $\boldsymbol{B} = (\boldsymbol{\beta}_1, \ldots, \boldsymbol{\beta}_{k-1})^{\mathrm{T}}$ is a $(k-1) \times p$ parameter matrix, and $\boldsymbol{\beta} = (\boldsymbol{\beta}_1^{\mathrm{T}}, \ldots, \boldsymbol{\beta}_{k-1}^{\mathrm{T}})^T$ is the vectorization of $\boldsymbol{B}$. For various choices for the penalty term $J(\cdot)$, we use different methods to solve the convex subproblems in DCA. First, we consider $J(\boldsymbol{f}) = \frac{1}{2}\sum_{q=1}^{k-1}\langle \boldsymbol{\beta}_q, \boldsymbol{\beta}_q \rangle$. In this case, we have that the DC components G and H as

$$G(\boldsymbol{\beta}) = \sum_{i=1}^{n} \left[\gamma H_{k-1}\{\langle \boldsymbol{B}\boldsymbol{x}_i, \boldsymbol{W}_{y_i} \rangle\} + (1-\gamma) \sum_{j \neq y_i} G_1\{\langle \boldsymbol{B}\boldsymbol{x}_i, \boldsymbol{W}_j \rangle\} \right] + \frac{\lambda}{2}\sum_{q=1}^{k-1}\langle \boldsymbol{\beta}_q, \boldsymbol{\beta}_q \rangle,$$
$$H(\boldsymbol{\beta}) = \sum_{i=1}^{n} \left[\gamma H_s\{\langle \boldsymbol{B}\boldsymbol{x}_i, \boldsymbol{W}_{y_i} \rangle\ \} + (1-\gamma) \sum_{j \neq y_i} G_s\{\langle \boldsymbol{B}\boldsymbol{x}_i, \boldsymbol{W}_j \rangle\} \right].$$

For DCA, note that the DC component H is a convex polyhedral function, thus for a point $\boldsymbol{\beta}^m$, a subgradient $\boldsymbol{g}^m \in \partial H(\boldsymbol{\beta}^m)(\boldsymbol{g}^m \in \mathbb{R}^{(k-1)p})$ can be computed efficiently. Define $\boldsymbol{g}_q^m$ to be the qth sub-vector in $\boldsymbol{g}^m$ that corresponds to $\boldsymbol{\beta}_q$. In particular, for the m-th iteration, we solve the following convex optimization

$$\min_{\boldsymbol{\beta}} \sum_{i=1}^{n} \left[\gamma H_{k-1}\{\langle \boldsymbol{B}\boldsymbol{x}_i, \boldsymbol{W}_{y_i} \rangle\} + (1-\gamma) \sum_{j \neq y_i} G_1\{\langle \boldsymbol{B}\boldsymbol{x}_i, \boldsymbol{W}_j \rangle\} \right] + \frac{\lambda}{2}\sum_{q=1}^{k-1}\langle \boldsymbol{\beta}_q, \boldsymbol{\beta}_q \rangle$$
$$- \sum_{q=1}^{k-1}\langle \boldsymbol{g}_q^m, \boldsymbol{\beta}_q \rangle. \tag{7}$$

We first introduce nonnegative slack variables $\boldsymbol{\xi} = (\xi_{11}, \xi_{12}, \ldots, \xi_{1k}, \ldots, \xi_{n1}, \xi_{n2}, \ldots, \xi_{nk})^{\mathrm{T}}$, and rewrite (7) as

$$\min_{\boldsymbol{\beta},\boldsymbol{\xi}} \sum_{i=1}^{n} \left\{ \gamma\xi_{iy_i} + (1-\gamma) \sum_{j \neq y_i} \xi_{ij} \right\} + \frac{\lambda}{2}\sum_{q=1}^{k-1}\langle \boldsymbol{\beta}_q, \boldsymbol{\beta}_q \rangle - \sum_{q=1}^{k-1}\langle \boldsymbol{g}_q^m, \boldsymbol{\beta}_q \rangle,$$
$$\text{s.t. } \xi_{iy_i} \geq k - 1 - \langle \boldsymbol{B}\boldsymbol{x}_i, \boldsymbol{W}_{y_i} \rangle;\ i = 1, \ldots, n,$$
$$\xi_{ij} \geq 1 + \langle \boldsymbol{B}\boldsymbol{x}_i, \boldsymbol{W}_j \rangle;\ i = 1, \ldots, n,\ j \neq y_i,$$

$$\xi_{ij} \geq 0;\ i = 1, \dots, n,\ j = 1, \dots, k.$$

Next, we introduce dual variables $\mu_{ij} \geq 0$ and $\alpha_{ij} \geq 0;\ i = 1, \dots, n,\ j = 1, \dots, k$, and the corresponding Lagrangian becomes

$$\begin{aligned}\mathbb{L}(\boldsymbol{\beta}, \boldsymbol{\xi}, \boldsymbol{\mu}, \boldsymbol{\alpha}) &= \sum_{i=1}^{n} \left\{ \gamma \xi_{iy_i} + (1-\gamma) \sum_{j \neq y_i} \xi_{ij} \right\} + \frac{\lambda}{2} \sum_{q=1}^{k-1} \langle \boldsymbol{\beta}_q, \boldsymbol{\beta}_q \rangle - \sum_{q=1}^{k-1} \langle \boldsymbol{g}_q^m, \boldsymbol{\beta}_q \rangle \\ &+ \sum_{i=1}^{n} \mu_{iy_i} (k - 1 - \langle \boldsymbol{B}\boldsymbol{x}_i, \boldsymbol{W}_{y_i} \rangle - \xi_{iy_i}) \\ &+ \sum_{i=1}^{n} \sum_{j \neq y_i} \mu_{ij} (1 + \langle \boldsymbol{B}\boldsymbol{x}_i, \boldsymbol{W}_j \rangle - \xi_{ij}) + \sum_{i=1}^{n} \sum_{j=1}^{k} \alpha_{ij} (-\xi_{ij}), \end{aligned} \tag{8}$$

where $\boldsymbol{\mu} = (\mu_{11}, \mu_{12}, \dots, \mu_{1k}, \dots, \mu_{n1}, \mu_{n2}, \dots, \mu_{nk})^{\mathrm{T}}$. After some calculation, one can show that $0 \leq \mu_{iy_i} \leq \gamma;\ i = 1, \dots, n$ and $0 \leq \mu_{ij} \leq 1 - \gamma;\ i = 1, \dots, n,\ j = 1, \dots, k,\ j \neq y_i$. Furthermore, we have that

$$\boldsymbol{\beta}_q = \frac{1}{\lambda} \left(\boldsymbol{g}_q^m + \sum_{i=1}^{n} \mu_{iy_i} W_{yi}^q \boldsymbol{x}_i - \sum_{i=1}^{n} \sum_{j \neq y_i} \mu_{ij} W_j^q \boldsymbol{x}_i \right), \tag{9}$$

where $W_{y_i}^q$ and W_j^q are the qth elements of $\boldsymbol{W}_{y_i}$ and $\boldsymbol{W}_j$, respectively. To simplify notation, we introduce $(k-1)$ matrices $\boldsymbol{X}^q;\ q = 1, \dots, k-1$, each with size $kn \times p$. For $\boldsymbol{X}^q$, its (ij)th row is defined to be $W_j^q \boldsymbol{x}_i^{\mathrm{T}}$ if $j \neq y_i$ and $-W_j^q \boldsymbol{x}_i^{\mathrm{T}}$ otherwise, for $i = 1, \dots, n,\ j = 1, \dots, k$. One can verify that (9) can be written as $\boldsymbol{\beta}_q = \frac{1}{\lambda}(\boldsymbol{g}_q^m - \boldsymbol{\mu}^{\mathrm{T}} \boldsymbol{X}^q)$, where $\boldsymbol{g}_q^m$ is the sub-vector in $\boldsymbol{g}^m$ that corresponds to $\boldsymbol{\beta}_q$. Substituting this into (8), we obtain the dual problem

$$\begin{aligned} &\min_{\boldsymbol{\mu}} \frac{1}{2} \boldsymbol{\mu}^{\mathrm{T}} \sum_{q=1}^{k-1} \boldsymbol{X}^q (\boldsymbol{X}^q)^{\mathrm{T}} \boldsymbol{\mu} - \lambda \langle \boldsymbol{\delta}, \boldsymbol{\mu} \rangle - \left\langle \sum_{q=1}^{k-1} \boldsymbol{X}^q (\boldsymbol{g}_q^m)^{\mathrm{T}}, \boldsymbol{\mu} \right\rangle, \\ &\text{s.t. } 0 \leq \mu_{iy_i} \leq \gamma;\ i = 1, \dots, n, \\ &0 \leq \mu_{ij} \leq 1 - \gamma;\ j \neq y_i,\ i = 1, \dots, n,\ j = 1, \dots, k, \end{aligned}$$

where $\boldsymbol{\delta} = (\delta_{11}, \delta_{12}, \cdots, \delta_{1k} m \cdots, \delta_{n1}, \delta_{n2}, \cdots, \delta_{nk})^{\mathrm{T}}$, and $\delta_{iy_i} = k - 1$, $\delta_{ij, j \neq y_i} = 1$. This dual problem is a quadratic programming problem with a positive semi-definite Hessian matrix and box-constraints. Due to the separability of the constraints, one can use the very efficient coordinate descent method to solve the optimization [15,41].

In order to perform variable selection, we consider the L_1 penalization with $J = \sum_{q=1}^{k-1} \|\boldsymbol{\beta}_q\|_1$. In this case, one can derive the following optimization problem in an analogous manner as (7),

$$\min_{\boldsymbol{\beta}} \sum_{i=1}^{n} \left[\gamma H_{k-1}\{\langle \boldsymbol{B}\boldsymbol{x}_i, \boldsymbol{W}_{y_i}\rangle\} + (1-\gamma) \sum_{j \neq y_i} G_1\{\langle \boldsymbol{B}\boldsymbol{x}_i, \boldsymbol{W}_j\rangle\} \right] + \lambda \sum_{q=1}^{k-1} \|\boldsymbol{\beta}_q\|_1 \tag{10}$$

$$- \sum_{q=1}^{k-1} \langle \boldsymbol{g}_q^m, \boldsymbol{\beta}_q \rangle.$$

To solve (10), we introduce nonnegative slack variables ξ_{ij}; $i = 1, \dots, n$, $j = 1, \dots, k$ and $\boldsymbol{\beta}^+$. Then (10) becomes a linear programming problem as follows:

$$\begin{aligned}
&\min_{\boldsymbol{\xi}, \boldsymbol{\beta}, \boldsymbol{\beta}^+} \sum_{i=1}^{n} \gamma \xi_{iy_i} + (1-\gamma) \sum_{i=1}^{n} \sum_{j \neq y_i} \xi_{ij} + \lambda \langle \boldsymbol{1}, \boldsymbol{\beta}^+ \rangle - \sum_{q=1}^{k-1} \langle \boldsymbol{g}_q^m, \boldsymbol{\beta}_q \rangle, \\
&\text{s.t. } \xi_{iy_i} \ge k - 1 - \langle \boldsymbol{B}\boldsymbol{x}_i, \boldsymbol{W}_{y_i} \rangle;\ i = 1, \dots, n, \\
&\xi_{ij} \ge 1 + \langle \boldsymbol{B}\boldsymbol{x}_i, \boldsymbol{W}_j \rangle;\ i = 1, \dots, n,\ j = 1, \dots, k,\ j \neq y_i, \\
&\boldsymbol{\beta}^+ - \boldsymbol{\beta} \ge \boldsymbol{0},\ \boldsymbol{\beta}^+ + \boldsymbol{\beta} \ge \boldsymbol{0},\ \boldsymbol{\beta}^+ \ge \boldsymbol{0},
\end{aligned}$$

which can be solved efficiently using existing linear programming software.

Lastly, we consider the mixed L_1 and L_2 penalty $\lambda J(\boldsymbol{f}) = \lambda_1 \sum_{q=1}^{k-1} \|\boldsymbol{\beta}_q\|_1 + \frac{\lambda_2}{2} \sum_{q=1}^{k-1} \langle \boldsymbol{\beta}_q, \boldsymbol{\beta}_q \rangle$. In this case, the optimization problem becomes

$$\begin{aligned}
\min_{\boldsymbol{\beta}} \Bigg(&\sum_{i=1}^{n} \left[\gamma H_{k-1}\{\langle \boldsymbol{B}\boldsymbol{x}_i, \boldsymbol{W}_{y_i}\rangle\} + (1-\gamma) \sum_{j \neq y_i} G_1\{\langle \boldsymbol{B}\boldsymbol{x}_i, \boldsymbol{W}_j\rangle\} \right] \\
&+ \lambda_1 \sum_{q=1}^{k-1} \|\boldsymbol{\beta}_q\|_1 + \frac{\lambda_2}{2} \sum_{q=1}^{k-1} \langle \boldsymbol{\beta}_q, \boldsymbol{\beta}_q \rangle - \sum_{q=1}^{k-1} \langle \boldsymbol{g}_q^m, \boldsymbol{\beta}_q \rangle \Bigg).
\end{aligned} \tag{11}$$

To solve (11), we employ the Alternating Linearization (AL, [17,28]) algorithm. In particular, we consider the following decomposition of (11),

$$\min_{\boldsymbol{\beta}} F(\boldsymbol{\beta}) = F_1(\boldsymbol{\beta}) + F_2(\boldsymbol{\beta}),$$

where F is the objective function in (11), F_1 corresponds to the robust MSVM loss term, and $F_2 = \frac{\lambda_2}{2} \sum_{q=1}^{k-1} \langle \boldsymbol{\beta}_q, \boldsymbol{\beta}_q \rangle - \sum_{q=1}^{k-1} \langle \boldsymbol{g}_q^m, \boldsymbol{\beta}_q \rangle + \lambda_1 \sum_{q=1}^{k-1} \|\boldsymbol{\beta}_q\|_1$. The idea of our AL algorithm is that, at each iteration, we solve two sub-problems, each consisting of one component, linearizations of the other components, and a proximal term. From these two sub-problems, we obtain two candidate updates for the original optimization. We then evaluate the objective function (11) at these two candidate updates, and if a candidate solution delivers an improvement, we update it for the original objective function.

The details of our AL method for (11) can be summarized as follows:

Alternating linearization algorithm

Initialization: $\bar{\boldsymbol{\beta}} \in \mathbb{R}^{(k-1)p}$ is the current solution vector, $\boldsymbol{\beta} \in \mathbb{R}^{(k-1)p}$ is the vector of unknown variables, and introduce $z_1, z_2, \boldsymbol{d}_1, \boldsymbol{d}_2 \in \mathbb{R}^{(k-1)p}$ initialized to zero vectors, and $\rho > 0$. In our description, we use $\boldsymbol{\beta}_q, \bar{\boldsymbol{\beta}}_q, \boldsymbol{d}_{1,q}, \boldsymbol{d}_{2,q}$ to denote the qth sub-vector of the corresponding vectors respectively, where $q = 1, 2, \ldots, k-1$.

Repeat

Sub-problem 1

$$z_1 = \underset{\boldsymbol{\beta}}{\operatorname{argmin}} \sum_{i=1}^{n} \gamma H_{k-1}\{\langle \boldsymbol{B}\boldsymbol{x}_i, \boldsymbol{W}_{y_i}\rangle\} + (1-\gamma)\sum_{j\neq y_i} G_1\{\langle \boldsymbol{B}\boldsymbol{x}_i, \boldsymbol{W}_j\rangle\}$$
$$+\sum_{q=1}^{k-1}\langle \boldsymbol{d}_{2,q}, \boldsymbol{\beta}_q\rangle + \frac{\rho}{2}\sum_{q=1}^{k-1}\|\boldsymbol{\beta}_q - \bar{\boldsymbol{\beta}}_q\|_2^2,$$
$$\boldsymbol{d}_1 = -\boldsymbol{d}_2 - \rho(z_1 - \bar{\boldsymbol{\beta}}).$$

If z_1 improves the objective function, set $\bar{\boldsymbol{\beta}} = z_1$.

Sub-problem 2

$$z_2 = \underset{\boldsymbol{\beta}}{\operatorname{argmin}} \frac{\lambda_2}{2}\sum_{q=1}^{k-1}\langle \boldsymbol{\beta}_q, \boldsymbol{\beta}_q\rangle - \sum_{q=1}^{k-1}\langle \boldsymbol{g}_q^m, \boldsymbol{\beta}_q\rangle + \lambda_1\sum_{q=1}^{k-1}\|\boldsymbol{\beta}_q\|_1$$
$$+\sum_{q=1}^{k-1}\langle \boldsymbol{d}_{1,q}, \boldsymbol{\beta}_q\rangle + \frac{\rho}{2}\sum_{q=1}^{k-1}\|\boldsymbol{\beta}_q - \bar{\boldsymbol{\beta}}_q\|_2^2,$$
$$\boldsymbol{d}_2 = -\boldsymbol{d}_1 - \rho(z_2 - \bar{\boldsymbol{\beta}}).$$

If z_2 improves the objective function, set $\bar{\boldsymbol{\beta}} = z_2$.

Until convergence.

Note that one can choose any $\rho > 0$ in the above algorithm. In our paper, we set $\rho = 1$. As a remark, we would like to point out that sub-problem 1 can be solved in an analogous manner as (7), using the coordinate descent algorithm. To make the algorithm more efficient, the solution of the previous iteration can be used as the starting point for the next iteration. Moreover, note that the L_1 and L_2 penalties are separable, therefore the sub-problem 2 has closed form solutions. This greatly enhances the computational speed of our method.

3.2 Algorithm for robust angle-based MSVM 2

For our second MSVM formulation (6), we can choose the DC components as

$$G(\boldsymbol{\beta}) = \sum_{i=1}^{n} H_1[\min_{j\neq y_i}\{\langle \boldsymbol{f}(\boldsymbol{x}_i), (\boldsymbol{W}_{y_i} - \boldsymbol{W}_j)\rangle\}] + \lambda J(\boldsymbol{f}),$$

and

$$H(\boldsymbol{\beta}) = \sum_{i=1}^{n} H_s[\min_{j \neq y_i}\{\langle \boldsymbol{f}(\boldsymbol{x}_i), (\boldsymbol{W}_{y_i} - \boldsymbol{W}_j)\rangle\}].$$

As in Sect. 3.1, we first consider the case when $J(\boldsymbol{f}) = \frac{1}{2}\sum_{q=1}^{k-1}\langle \boldsymbol{\beta}_q, \boldsymbol{\beta}_q\rangle$. In the mth iteration of DCA, we solve the following optimization problem

$$\min_{\boldsymbol{\beta}} \left(\sum_{i=1}^{n} H_1[\min_{j \neq y_i}\{\langle \boldsymbol{B}\boldsymbol{x}_i, (\boldsymbol{W}_{y_i} - \boldsymbol{W}_j)\rangle\}] + \frac{\lambda}{2}\sum_{q=1}^{k-1}\langle \boldsymbol{\beta}_q, \boldsymbol{\beta}_q\rangle - \sum_{q=1}^{k-1}\langle \boldsymbol{g}_q^m, \boldsymbol{\beta}_q\rangle \right). \quad (12)$$

Using nonnegative slack variables $\xi_i,\ i = 1, \ldots, n$, (12) is equivalent to

$$\begin{aligned} &\min_{\boldsymbol{\beta},\boldsymbol{\xi}} \sum_{i=1}^{n}\xi_i + \frac{\lambda}{2}\sum_{q=1}^{k-1}\langle \boldsymbol{\beta}_q, \boldsymbol{\beta}_q\rangle - \sum_{q=1}^{k-1}\langle \boldsymbol{g}_q^m, \boldsymbol{\beta}_q\rangle, \\ &\text{s.t. } \ \xi_i \geq 1 - \min_{j \neq y_i}\langle \boldsymbol{B}\boldsymbol{x}_i, (\boldsymbol{W}_{y_i} - \boldsymbol{W}_j)\rangle; i = 1, \ldots, n, \\ &\quad \xi_i \geq 0; i = 1, \ldots, n. \end{aligned}$$

Next, by introducing nonnegative dual variables $\boldsymbol{\mu} \in \mathbb{R}^{(k-1)n}$ and $\boldsymbol{\alpha} \in \mathbb{R}^n$, we have the corresponding Lagrangian

$$\begin{aligned} \mathbb{L}(\boldsymbol{\beta}, \boldsymbol{\xi}, \boldsymbol{\mu}, \boldsymbol{\alpha}) = &\sum_{i=1}^{n}\xi_i + \frac{\lambda}{2}\sum_{q=1}^{k-1}\langle \boldsymbol{\beta}_q, \boldsymbol{\beta}_q\rangle - \sum_{q=1}^{k-1}\langle \boldsymbol{g}_q^m, \boldsymbol{\beta}_q\rangle \\ &+ \sum_{i=1}^{n}\sum_{j \neq y_i}\mu_{ij}(1 - \langle \boldsymbol{B}\boldsymbol{x}_i, \boldsymbol{W}_{y_i} - \boldsymbol{W}_j\rangle - \boldsymbol{\xi}_i) + \sum_{i=1}^{n}\alpha_i(-\xi_i). \quad (13) \end{aligned}$$

With some calculation, one can show that for a fixed i, $\sum_{j \neq y_i}\mu_{ij} \leq 1$. Furthermore, we have that

$$\boldsymbol{\beta}_q = \frac{1}{\lambda}\left(\boldsymbol{g}_q^m + \sum_{i=1}^{n}\sum_{j \neq y_i}\mu_{ij}(W_{y_i}^q - W_j^q)\boldsymbol{x}_i \right), \quad (14)$$

where $\boldsymbol{\beta}_q;\ q = 1, \ldots, k-1$, is the parameter vector of the qth classification function, and W_j^q is the q-th entry of $\boldsymbol{W}_j$.

Similar to Sect. 3.1, we define $k-1$ matrices $\boldsymbol{X}^1, \ldots, \boldsymbol{X}^{k-1}$ with dimension $(k-1)n \times p$ to simplify notation. In particular, for each q, and let the ijth row of $\boldsymbol{X}^q$ be $(W_j^q - W_{y_i}^q)\boldsymbol{x}_i^{\mathrm{T}};\ i = 1, \ldots, n,\ \ j = 1, \ldots, k,\ \ j \neq y_i$. One can verify that (14) can be written as $\boldsymbol{\beta}_q = \frac{1}{\lambda}(\boldsymbol{g}_q^m - \boldsymbol{\mu}^{\mathrm{T}}\boldsymbol{X}^q)$. By substituting this into (13), we obtain the dual problem

$$\min_{\boldsymbol{\mu}} \frac{1}{2}\boldsymbol{\mu}^{\mathrm{T}} \sum_{q=1}^{k-1} \boldsymbol{X}^q (\boldsymbol{X}^q)^{\mathrm{T}} \boldsymbol{\mu} - \lambda\langle \mathbf{1}, \boldsymbol{\mu}\rangle - \left\langle \sum_{q=1}^{k-1} \boldsymbol{X}^q (\boldsymbol{g}_q^m)^{\mathrm{T}}, \boldsymbol{\mu} \right\rangle,$$
$$s.t. \sum_{j\neq y_i} \mu_{ij} \leq 1;\ i = 1, \ldots, n,$$
$$\boldsymbol{\mu} \geq \mathbf{0}. \tag{15}$$

The optimization problem (15) is a quadratic programming problem with box constraints and linear inequalities. Due to the block separability structure of the linear inequality constraints, we can solve this problem by iterating through each block [36,41]. In particular, for a fixed i, let $\boldsymbol{Q} = \sum_{q=1}^{k-1} \boldsymbol{X}^q(\boldsymbol{X}^q)^{\mathrm{T}}$, and let $\boldsymbol{Q}_i$ be the sub-matrix corresponding to μ_{ij}, $j \neq y_i$. One can verify that the sub-problem is a small scale positive semi-definite quadratic programming problem with box constraints and a single inequality constraint

$$\min \frac{1}{2}\boldsymbol{\mu}_{i,j\neq y_i}^{\mathrm{T}} \boldsymbol{Q}_i \boldsymbol{\mu}_{i,j\neq y_i} + \boldsymbol{v}^{\mathrm{T}}\boldsymbol{\mu}_{i,j\neq y_i},$$
$$s.t. \sum_{j\neq y_i} \mu_{ij} \leq 1,\ \mu_{ij} \geq 0;\ j = 1, \ldots, k,\ j \neq y_i,$$

where $\boldsymbol{v}$ is a vector that depends only on λ, $\boldsymbol{X}^q$ and $\boldsymbol{g}^m$. When $k = 3$, the problem is a simple quadratic function with two variables and the solutions can be easily obtained. For $k > 3$, the sub-problem can be solved efficiently using the coordinate descent algorithm [38].

In the cases of L_1 and mixed $L_1 + L_2$ penalties, one can employ similar strategies as in Sect. 3.1, and we omit the details here.

4 Statistical properties

In this section, we explore some statistical properties of our proposed classifiers using loss functions (3) and (6). In particular, we first establish the Fisher consistency of (3) and (6), then show how to estimate the future prediction error rate using the training dataset.

Fisher consistency was introduced in Sect. 2.2. The next theorem shows that the proposed robust angle-based MSVM can enjoy Fisher consistency with appropriately selected parameters s and γ.

Theorem 1 *Our first robust angle-based MSVM (3) is Fisher consistent with $\gamma \in [0, 1/2]$ and $s \leq 0$, and our second robust angle-based MSVM (6) is Fisher consistent with $s \in [-\frac{1}{k-1}, 0]$.*

According to Theorem 4, with infinitely many training observations and an appropriate $\mathscr{F}$, our robust angle-based MSVMs can achieve the best classification accuracy.

In practice, it is desirable to study the prediction performance of the obtained classifier on future testing data, in terms of $E[I\{Y \neq \hat{y}_{\hat{f}}(\boldsymbol{X})\}]$. To this end, one possible approach is to use the empirical prediction error rate on the training dataset, $\frac{1}{n}\sum_{i=1}^{n} I\{y_i \neq \hat{y}_{\hat{f}}(\boldsymbol{x}_i)\}$, to estimate the future error rate. However, it is well known that the empirical error rate often underestimates $E[I\{Y \neq \hat{y}_{\hat{f}}(\boldsymbol{X})\}]$. The next theorem shows that for our new classifiers, the future error rate exceeds the empirical rate by a small amount that can converge to zero at a fast rate. Here we use linear learning with L_1 or L_2 penalties, and reproducing kernel Hilbert space learning [44] as examples. With a little abuse of notation, we denote by $\hat{\boldsymbol{f}}$ the solution to our new MSVM methods (3) or (6).

Theorem 2 *The solution $\hat{\boldsymbol{f}}$ to (3) or (6) satisfies that, with probability at least $1-\delta$,*

$$E[I\{Y \neq \hat{y}_{\hat{f}}(\boldsymbol{X})\}] \leq \left[\frac{1}{n}\sum_{i=1}^{n} I\{y_i \neq \hat{y}_{\hat{f}}(\boldsymbol{x}_i)\} + 3\{\log(2/\delta)/n\}^{1/2} + Z\right], \qquad (16)$$

where

$$Z = \begin{cases} \frac{C_1}{\lambda}\sqrt{\frac{\log(p)}{n}} & \text{for linear learning with} L_1 \text{penalty,} \\ \frac{C_2}{\lambda^{1/2}}\sqrt{\frac{p}{n^{1/2}}} & \text{for linear learning with} L_2 \text{penalty,} \\ \frac{C_3}{\lambda\sqrt{n}} & \text{for RKHS learning with a separable kernel,} \end{cases}$$

and C_1, C_2, C_3 are constants (specified in the “Appendix”) that do not depend on n or p.

Theorem 2 shows that for our robust MSVM classifiers, one can obtain an upper bound for the future prediction error rate using the empirical error rate, and the corresponding difference can converge to zero at a fast rate. For example, as $n \to \infty$, we can let $\lambda \to 0$ (as is typical for general machine learning problems) at the rate of $O\{\log(n)^{-1/2}\}$. In this case, we have that $Z = O_P\{\sqrt{\log(p)\log(n)/n}\}$ for L_1 learning, $Z = O_P\{\sqrt{p\log(n)^{1/2}/n^{1/2}}\}$ for L_2 learning, and $Z = O_P\{\sqrt{\log(n)/n}\}$ for kernel learning.

As a remark, we would like to point out that Theorem 2 also reveals the effectiveness of L_1 penalized methods over the L_2 ones on practical problems where the underlying signal is sparse. To see this, consider a problem where the classification signal depends only on a handful predictors, and the remaining predictors are noise variables. In this case, it is well known that L_1 penalized methods can deliver parsimonious classifiers that can fit the data well [45]. On the other hand, L_2 penalized classifiers cannot deliver sparse classification models. Because we can typically expect a small empirical error rate on training datasets, Theorem 2 shows that when the dimension p is ultrahigh, the corresponding convergence rate of the difference term Z can be much faster in the L_1 case. In other words, the L_1 penalized classifiers can perform much better in terms of future prediction accuracy. This is consistent with many existing experiments in the literature.

5 Numerical results

In this section, we investigate the performance of our proposed robust angle-based SVMs using simulated and real datasets.

5.1 Simulated experiments

In this section, we conduct a simulation example to demonstrate the numerical performance of our new classifiers. We use 100 observations for training, and select the best regularization parameters via a grid search on a separate tuning dataset with 1000 observations. Then, we evaluate the prediction performance of the obtained classifiers on a testing dataset with size 10,000. We compare the performance of RMSVM by [32], MSVM by [31], and our proposed RSVMs (3) and (6).

We generate the simulated datasets as follows. We choose $k = 3$ with $\mathrm{pr}(Y = j) = 1/3$ for $j = 1, 2, 3$. Among the observations in the training and tuning sets, we generate 5 and 10% outliers (denoted by perc = 5% and perc = 10%) to contaminate the data. The classification signal depends only on two predictors X_1 and X_2. For the uncontaminated observations, the marginal distribution of $(X_1, X_2)^{\mathrm{T}}$ for $Y = j$ is $N(\mu_j, 0.16\boldsymbol{I}_2)$, where $\boldsymbol{I}_2$ is the identity matrix, and $\boldsymbol{\mu}_j$'s are equally distributed on the unit circle with $\boldsymbol{\mu}_1 = (1, 0)^{\mathrm{T}}$. For the contaminated observations, the marginal distribution of $(X_1, X_2)^{\mathrm{T}}$ is $N((3, 0)^{\mathrm{T}}, 0.16\boldsymbol{I}_2)$ for $Y = 1, 2, 3$. Moreover, we generate noise variables to make the dimension p higher. In particular, the noise predictors are generated independently from a uniform distribution on $[-1, 1]$. We report the results of $p = 2, 20, 200$.

We compare the performance of our two robust angle-based SVMs (RSVMs 1–2) (3) and (6), with truncation at $s = 0$ (denoted by T_0) and $s = -1/(k-1)$ (denoted by T_K). For the regularization terms, we employ three different penalty functions, namely, the L_2, L_1, and $L_1 + L_2$ penalties. In particular, the candidate sets of L_1 and L_2 tuning parameters consist of 30 candidates each, and the candidate sets of $L_1 + L_2$ tuning parameters consist of $30 \times 30 = 900$ different values. For each setting of the simulation, we run the experiments 100 times. We report some of the classification results in Fig. 3. Full numerical results of the simulated studies are reported in Tables 1, 2 and 3. From Fig. 3 and the tables, one can see that our proposed RSVMs perform better than the original SVMs. Furthermore, our methods with truncation at $s = 0$ and $s = -1/(k-1)$ perform similarly. To avoid intensive tuning of s and to obtain more stable results, we recommend to use $s = -1/(k-1)$ in practice.

Note that only the first 2 predictors contain useful classification signals, and the remaining are noise variables. In Table 4, we report the average numbers of selected variables for different classifiers and penalties with $p = 200$. The L_2 regularization chooses all variables in the resulting classifiers, while the L_1 and $L_1 + L_2$ penalties lead to more parsimonious classifiers. Note that all methods with L_1 and $L_1 + L_2$ penalties are able to identify the underlying two predictors. In terms of classification error rates, the two sparse penalties perform better than the pure L_2 method, with $L_1 + L_2$ slightly better than L_1.

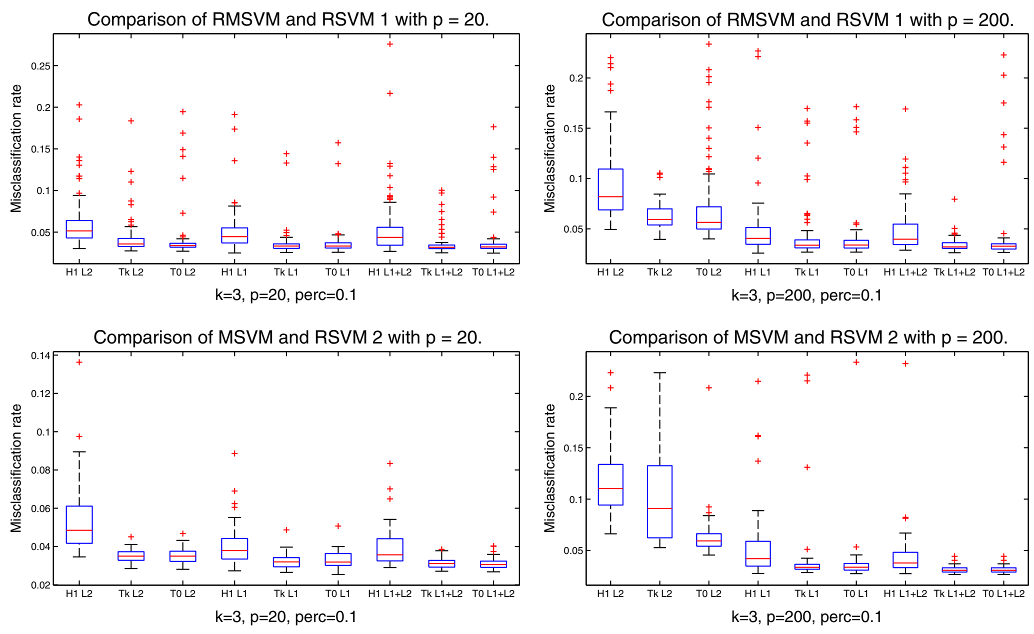

Fig. 3 Boxplots of misclassification rates for RMSVM (denoted by H_1 on the top panels), our RSVM 1 with truncation at $s = -1/(k-1)$ (T_K on the top panels) and at $s = 0$ (T_0 on the top panels), MSVM (denoted by H_1 on the bottom panels), and our RSVM 2 with truncation at $s = -1/(k-1)$ (T_K on the bottom panels) and at $s = 0$ (T_0 on the bottom panels) on the simulated example

Table 1 Classification error rates for RMSVM, MSVM and our RSVMs 1–2 on the simulated data with $k = 3$, $p = 2$

Penalty		L_2	L_1	$L_1 + L_2$
Perc	Loss	Error	Error	Error
5%	RMSVM	$3.3600_{(0.0062)}$	$3.5575_{(0.0063)}$	$3.3990_{(0.0069)}$
	T_0 RSVM 1	$3.1159_{(0.0034)}$	$3.3125_{(0.0057)}$	$3.0756_{(0.0030)}$
	T_k RSVM 1	$3.1148_{(0.0043)}$	$3.2874_{(0.0045)}$	$3.0968_{(0.0028)}$
	MSVM	$3.2572_{(0.0032)}$	$3.4097_{(0.0049)}$	$3.2130_{(0.0031)}$
	T_0 RSVM 2	$3.0654_{(0.0023)}$	$3.2780_{(0.0036)}$	$3.0446_{(0.0027)}$
	T_k RSVM 2	$3.0766_{(0.0021)}$	$3.2346_{(0.0031)}$	$3.0123_{(0.0023)}$
10%	RMSVM	$4.6710_{(0.0237)}$	$5.0689_{(0.0202)}$	$4.6662_{(0.0250)}$
	T_0 RSVM 1	$4.7892_{(0.0649)}$	$3.8702_{(0.0241)}$	$3.3646_{(0.0180)}$
	T_k RSVM 1	$3.4628_{(0.0098)}$	$3.7674_{(0.0174)}$	$3.4748_{(0.0180)}$
	MSVM	$4.3474_{(0.0214)}$	$4.1759_{(0.0135)}$	$4.5220_{(0.0209)}$
	T_0 RSVM 2	$3.2404_{(0.0194)}$	$3.2075_{(0.0031)}$	$2.9990_{(0.0023)}$
	T_k RSVM 2	$3.2451_{(0.0170)}$	$3.1844_{(0.0029)}$	$2.9890_{(0.0022)}$

Here Perc stands for the percentage of data that are contaminated

As a measurement of computational speeds, for the simulated example with $p = 200$, we report the average running time of one replication for the compared methods in Table 5. One can see that since the optimization problems are more involved, our

Table 2 Classification error rates for RMSVM, MSVM and our RSVMs 1–2 on the simulated data with $k = 3$, $p = 20$

Penalty		L_2	L_1	$L_1 + L_2$
Perc	Loss	Error	Error	Error
5%	RMSVM	$4.0991_{(0.0089)}$	$3.4977_{(0.0072)}$	$3.4074_{(0.0047)}$
	T_0 RSVM 1	$3.4220_{(0.0103)}$	$3.4978_{(0.0055)}$	$3.1792_{(0.0032)}$
	T_k RSVM 1	$3.6043_{(0.0050)}$	$3.3798_{(0.0047)}$	$3.2436_{(0.0037)}$
	MSVM	$3.9090_{(0.0063)}$	$3.3828_{(0.0046)}$	$3.3400_{(0.0052)}$
	T_0 RSVM 2	$3.4982_{(0.0041)}$	$3.3873_{(0.0043)}$	$3.1500_{(0.0025)}$
	T_k RSVM 2	$3.4442_{(0.0037)}$	$3.3396_{(0.0035)}$	$3.0922_{(0.0021)}$
10%	RMSVM	$4.0991_{(0.0140)}$	$4.6318_{(0.0172)}$	$5.0172_{(0.0257)}$
	T_0 RSVM 1	$3.5752_{(0.0055)}$	$3.4000_{(0.0051)}$	$3.3710_{(0.0116)}$
	T_k RSVM 1	$3.4733_{(0.0044)}$	$3.3798_{(0.0049)}$	$3.4596_{(0.0084)}$
	MSVM	$5.5827_{(0.0194)}$	$4.6140_{(0.0252)}$	$4.1241_{(0.0149)}$
	T_0 RSVM 2	$3.5240_{(0.0041)}$	$3.3520_{(0.0042)}$	$3.1598_{(0.0031)}$
	T_k RSVM 2	$3.5147_{(0.0039)}$	$3.4888_{(0.0204)}$	$3.4147_{(0.0034)}$

Here Perc stands for the percentage of data that are contaminated

Table 3 Classification error rates for RMSVM, MSVM and our RSVMs 1–2 on the simulated data with $k = 3$, $p = 200$

Penalty		L_2	L_1	$L_1 + L_2$
Perc	Loss	Error	Error	Error
5%	RMSVM	$6.3898_{(0.0121)}$	$3.4861_{(0.0098)}$	$3.5316_{(0.0077)}$
	T_0 RSVM 1	$5.5830_{(0.0093)}$	$3.5360_{(0.0063)}$	$3.3142_{(0.0039)}$
	T_k RSVM 1	$5.7684_{(0.0085)}$	$3.3287_{(0.0045)}$	$3.2944_{(0.0040)}$
	MSVM	$7.8186_{(0.0250)}$	$3.4781_{(0.0081)}$	$3.5385_{(0.0131)}$
	T_0 RSVM 2	$5.9862_{(0.0093)}$	$3.3206_{(0.0073)}$	$3.3900_{(0.0061)}$
	T_k RSVM 2	$6.3608_{(0.0183)}$	$3.4235_{(0.0040)}$	$3.2542_{(0.0051)}$
10%	RMSVM	$9.1100_{(0.0410)}$	$5.1094_{(0.0309)}$	$5.2558_{(0.0292)}$
	T_0 RSVM 1	$5.7400_{(0.0241)}$	$3.4894_{0.0052)}$	$3.5366_{(0.0127)}$
	T_k RSVM 1	$6.6500_{(0.0362)}$	$3.9901_{(0.0247)}$	$3.4498_{(0.0100)}$
	MSVM	$12.0516_{(0.0364)}$	$5.3859_{(0.0346)}$	$4.1383_{(0.0103)}$
	T_0 RSVM 2	$6.5674_{(0.0300)}$	$3.8189_{(0.0275)}$	$3.1750_{(0.0026)}$
	T_k RSVM 2	$10.1318_{(0.0467)}$	$4.0308_{(0.0334)}$	$2.9816_{(0.0013)}$

Here Perc stands for the percentage of data that are contaminated

RSVMs with the DCA take longer to compute, compared with RMSVM and MSVM without truncation. However, they are still relatively efficient to implement. Note that the candidate set of tuning parameters for the $L_1 + L_2$ penalty is much larger than that of L_1 or L_2 penalties, hence the running time of the methods with the $L_1 + L_2$ penalty is longer.

Table 4 The number of predictors selected by RMSVM, MSVM and our RSVMs 1–2 on the simulated example with $p = 200$ and data contamination percentage 10%

Penalty	L_2	L_1	$L_1 + L_2$
RMSVM	200.0000	3.9800	8.4000
T_0 RSVM 1	200.0000	6.0200	27.8000
T_K RSVM 1	200.0000	6.1400	6.8000
MSVM	200.0000	6.9200	3
T_0 RSVM 2	200.0000	5.4000	3
T_K RSVM 2	200.0000	5.1600	3

Table 5 Running time of one replication for various classifiers and penalty functions in the simulated example with $p = 200$

Penalty	L_2 Running time (s)	L_1 Running time (s)	$L_1 + L_2$ Running time (s)
RMSVM	0.3736	12.9099	152.0332
T_0 RSVM 1	4.1636	31.7880	717.1089
T_k RSVM 1	3.8809	26.6204	674.5778
MSVM	2.2035	11.4896	1347.3421
T_0 RSVM 2	8.5175	22.9063	4208.3969
T_k RSVM 2	6.4516	16.9095	2599.3510

5.2 Real data analysis

In this section, we investigate the performance of our proposed classifiers using a real application dataset Small Round Blue Cell Tumors (SRBCT, [11]), which can be found on the UCI Machine Learning Repository [2]. The SRBCT dataset consists of 4 different types of children brain tumors, including Ewing sarcoma (EWS), neuroblastoma (NB), rhabdomyosarcoma (RMS), and Burkitt's Lymphoma (BL). Because treatment options and responses to therapy vary widely depending on the diagnosis, accurate prediction of the SRBCT subtype is highly desirable. However, these cancer subtypes are similar in routine histology and are difficult to differentiate using regular clinical results. The dataset contains gene expression levels of 2308 genes, and there are 83 patients in total. In particular, we have 29 EWSs, 18 NBs, 25 RMSs, and 11 BLs. In our analysis, we split the data into 3 equal parts for training, tuning, and testing. We first compare the performance of RMSVM [32], MSVM by [31], and our proposed RSVMs (3) and (6). Here, we choose the truncation locations in the same way as in Sect. 5.1. We also contaminate the dataset with outliers. In particular, we choose 10% of the observations, randomly relabel them into another class, and add a constant 2 to each of the corresponding predictors. Then we run the examples again using RMSVM, MSVM and our proposed RSVMs 1–2 to demonstrate the effect of outliers on classification accuracy.

We report the average misclassification error rates over 50 replicates for various methods in Table 6. We also report the running time in seconds for each method in Table 7. The running time for the method with $L_1 + L_2$ penalty is higher since the

Table 6 Classification error rates for RMSVM, MSVM and our RSVMs 1–2 on the SRBCT dataset

Penalty		L_2	L_1	$L_1 + L_2$
Perc	Loss	Error	Error	Error
0%	RMSVM	$3.6000_{(0.0542)}$	$6.1333_{(0.0795)}$	$4.3859_{(0.0705)}$
	T_0 RSVM 1	$3.8666_{(0.0540)}$	$6.2666_{(0.0801)}$	$3.2894_{(0.0638)}$
	T_k RSVM 1	$3.3333_{(0.0526)}$	$6.0000_{(0.0800)}$	$4.3859_{(0.0705)}$
	MSVM	$5.3333_{(0.0602)}$	$7.2000_{(0.0850)}$	$3.4736_{(0.0482)}$
	T_0 RSVM 2	$5.3333_{(0.0602)}$	$7.2000_{(0.0850)}$	$3.0526_{(0.0464)}$
	T_k RSVM 2	$5.3333_{(0.0602)}$	$7.2000_{(0.0850)}$	$3.4736_{(0.0482)}$
10%	RMSVM	$6.9333_{(0.0830)}$	$11.0666_{(0.1171)}$	$6.0000_{(0.0728)}$
	T_0 RSVM 1	$4.8000_{(0.0539)}$	$8.5333_{(0.0852)}$	$4.8000_{(0.0539)}$
	T_k RSVM 1	$6.1333_{(0.0671)}$	$9.0666_{(0.0891)}$	$5.3333_{(0.0571)}$
	MSVM	$8.8000_{(0.0743)}$	$10.4000_{(0.0843)}$	$5.1714_{(0.0670)}$
	T_0 RSVM 2	$7.4666_{(0.0695)}$	$9.2000_{(0.0840)}$	$3.7838_{(0.0559)}$
	T_k RSVM 2	$8.4000_{(0.0748)}$	$10.4000_{(0.0821)}$	$4.0390_{(0.0545)}$

Here Perc stands for the percentage of data that are contaminated

Table 7 Running time in seconds for RMSVM, MSVM and our RSVMs 1–2 on the SRBCT dataset

Penalty		L_2	L_1	$L_1 + L_2$
Perc	Loss	Running time(s)	Running time(s)	Running time (s)
0%	RMSVM	0.7095	78.7449	474.5619
	T_0 RSVM 1	4.2161	88.7830	2770.0465
	T_k RSVM 1	4.0069	86.0682	2833.6601
	MSVM	0.5916	1292.2680	716.5681
	T_0 RSVM 2	1.8748	3409.9990	3348.6898
	T_k RSVM 2	3.6891	3497.6771	3260.9417
10%	RMSVM	3.6029	84.5760	596.2820
	T_0 RSVM 1	14.7556	111.6447	3053.0931
	T_k RSVM 1	14.7022	104.4469	3198.6766
	MSVM	0.5898	1639.3733	685.2144
	T_0 RSVM 2	3.8194	5383.0758	2730.4939
	T_k RSVM 2	4.6654	5169.2126	2663.3808

Here Perc stands for the percentage of data that are contaminated

number of tuning parameters is much bigger. Note that when Perc $= 0\%$, the results of MSVM and RSVMs 1–2 using L_1 or L_2 penalties are the same. This is because all these methods have perfect prediction on the training set, therefore no truncation is needed. From the results, we can see that our proposed methods with truncations are very competitive in terms of classification accuracy, especially when the data are contaminated with outliers. For this dataset, among 3 types of penalties, $L_1 + L_2$ works the best. This can be due to the grouping effects of this penalty which allows highly correlated input variables, commonly in genetic data, to have similar fitted coefficients and simultaneously allow automatic variable selection and shrinkage [46]. Between two choices of truncated location, $s = -1/(k-1)$ works slightly better than $s = 0$.

In terms of variable selection, even when 10% of the data are contaminated, our proposed methods with the $L_1 + L_2$ penalty are able to identify important signature genes that can be used to predict SRBCT subtypes. In particular, we identify genes with ID 770394 and 1435862 that are known to be associated with EWS, genes 812105 and 325182 associated with NB, gene 796258 associated with RMS, and gene 183337 associated with BL [11]. Here, genes 770394 and 1435862 appear in the final classifier for all the repetitions, whereas gene 183337 is selected 47 times out of 50 repetitions (94%), partly because the number of samples of this tumor subtype is small.

6 Discussion

In this paper, we consider how to alleviate the effect of outliers in the angle-based classification framework. The existing angle-based methods impose heavy loss values on outliers, hence the resulting classifiers can be unstable and suboptimal. To overcome this difficulty, we employ truncated hinge loss functions, and propose two robust angle-based MSVM methods. Because the corresponding optimization problems are non-convex, we use the DCA to solve the corresponding optimization, and develop the algorithms for our RSVMs using various penalty functions. Theoretical results, including Fisher consistency and prediction error bounds are obtained. Numerical results with both simulated and real data demonstrate that our new classifiers are very competitive, especially at the presence of outliers in the data. One interesting future research direction is to further understand the solutions obtained by DCA for our two proposed robust angle-based MSVMs, along the line of the recent work by [37].

Acknowledgements The authors would like to thank the reviewers and editors for their helpful comments and suggestions which led to a much improved presentation. Yufeng Liu's research was supported in part by National Science Foundation Grant IIS1632951 and National Institute of Health Grant R01GM126550. Chong Zhang's research was supported in part by National Science and Engineering Research Council of Canada (NSERC). Pham was supported in part by National Science Foundation Grant DMS1127914 and the Hobby Postdoctoral Fellowship.

Appendix

Proof of Theorem 4 To prove the theorem, we need the following lemma, of which the proof can be found in [49].

Lemma 1 ([49], Lemma 1) *Suppose we have an arbitrary* $\boldsymbol{f} \in \mathbb{R}^{k-1}$. *For any* $u, v \in \{1, \ldots, k\}$ *such that* $u \neq v$, *define* $\boldsymbol{T}_{u,v} = \boldsymbol{W}_u - \boldsymbol{W}_v$. *For any scalar* $z \in \mathbb{R}$, $\langle(\boldsymbol{f} + z\boldsymbol{T}_{u,v}), \boldsymbol{W}_w\rangle = \langle \boldsymbol{f}, \boldsymbol{W}_w\rangle$, *where* $w \in \{1, \ldots, k\}$ *and* $w \neq u, v$. *Furthermore, we have that* $\langle(\boldsymbol{f} + z\boldsymbol{T}_{u,v}), \boldsymbol{W}_u\rangle - \langle \boldsymbol{f}, \boldsymbol{W}_u\rangle = -\langle(\boldsymbol{f} + z\boldsymbol{T}_{v,u}), \boldsymbol{W}_v\rangle + \langle \boldsymbol{f}, \boldsymbol{W}_v\rangle$.

We first prove that the loss function (3) is Fisher consistent with $\gamma \leq 1/2$ and $s \leq 0$. Assume that, without loss of generality, $P_1 > P_2 \geq \cdots \geq P_k$. The goal is to show that $\langle \boldsymbol{f}^*, \boldsymbol{W}_1\rangle > \langle \boldsymbol{f}^*, \boldsymbol{W}_j\rangle$ for any $j \neq 1$. Inspired by the proof in [32,47], we complete our proof with four steps.

The first step is to show that there exists a theoretical minimizer $\boldsymbol{f}^*$ such that $\langle \boldsymbol{f}^*, \boldsymbol{W}_j\rangle \le k-1$ for all j. Otherwise, suppose $\langle \boldsymbol{f}^*, \boldsymbol{W}_i\rangle > k-1$ for some i, there must exist $q \in \{1,\dots,k\}$, $q \neq i$, such that $\langle \boldsymbol{f}^*, \boldsymbol{W}_q\rangle < -1$, by the sum-to-zero property of the inner products $\sum_{j=1}^{k}\langle \boldsymbol{f}^*, \boldsymbol{W}_j\rangle = 0$. Now one can decrease $\langle \boldsymbol{f}^*, \boldsymbol{W}_i\rangle$ by a small amount, and increase $\langle \boldsymbol{f}^*, \boldsymbol{W}_q\rangle$ by the same amount (Lemma 1), and the variation of the conditional loss depends on s. Specifically, if $s < -\frac{1}{k-1}$, the conditional loss is decreased, which is a contradiction to the definition of $\boldsymbol{f}^*$. If $-\frac{1}{k-1} \le s \le 0$, the conditional loss remains the same. However, one can keep doing this till all $\langle \boldsymbol{f}^*, \boldsymbol{W}_j\rangle \le k-1$, which is a contradiction to the assumption $\langle \boldsymbol{f}^*, \boldsymbol{W}_i\rangle > k-1$.

The second step is to show that $\langle \boldsymbol{f}^*, \boldsymbol{W}_1\rangle \ge \langle \boldsymbol{f}^*, \boldsymbol{W}_j\rangle$ for any $j \neq 1$ using contradiction. Suppose $\langle \boldsymbol{f}^*, \boldsymbol{W}_1\rangle < \langle \boldsymbol{f}^*, \boldsymbol{W}_i\rangle$ for some i. By the definition of $S(\boldsymbol{x}) = E[\phi_1\{\boldsymbol{f}(\boldsymbol{X}), Y\} \mid \boldsymbol{X} = \boldsymbol{x}]$, we can simplify it as

$$S(\boldsymbol{x}) \triangleq \sum_{j=1}^{k} P_j h_\gamma(\langle \boldsymbol{f},\ \boldsymbol{W}_j\rangle) + (1-\gamma)\sum_{j=1}^{k} R_s(\langle \boldsymbol{f},\ \boldsymbol{W}_j\rangle).$$

where $h_\gamma(u) = \gamma T_{(k-1)s}(u) - (1-\gamma)R_s(u)$. Because $h_\gamma(u)$ is monotone decreasing for any $0 \le \gamma \le 1$, we have $h_\gamma(\langle \boldsymbol{f}^*,\ \boldsymbol{W}_1\rangle) \ge h_\gamma(\langle \boldsymbol{f}^*,\ \boldsymbol{W}_i\rangle)$. We claim that $h_\gamma(\langle \boldsymbol{f}^*,\ \boldsymbol{W}_1\rangle) \le h_\gamma(\langle \boldsymbol{f}^*,\ \boldsymbol{W}_j\rangle)$ for all $j \neq 1$. If it is not true, there must exist $h_\gamma(\langle \boldsymbol{f}^*,\ \boldsymbol{W}_1\rangle) > h_\gamma(\langle \boldsymbol{f}^*,\ \boldsymbol{W}_j\rangle)$ for some j. Then we can define $\boldsymbol{f}'(\boldsymbol{x}) \in \mathbb{R}^{k-1}$ such that $\langle \boldsymbol{f}^*, \boldsymbol{W}_1\rangle = \langle \boldsymbol{f}', \boldsymbol{W}_j\rangle$ and $\langle \boldsymbol{f}^*, \boldsymbol{W}_j\rangle = \langle \boldsymbol{f}', \boldsymbol{W}_1\rangle$ (the existence of such $\boldsymbol{f}'$ is guaranteed by Lemma 1). One can verify that $\boldsymbol{f}'$ is the minimizer of $S(\boldsymbol{x})$, not $\boldsymbol{f}^*$. This contradicts with the definition of $\boldsymbol{f}^*$. Therefore, we obtain $h_\gamma(\langle \boldsymbol{f}^*,\ \boldsymbol{W}_1\rangle) = h_\gamma(\langle \boldsymbol{f}^*,\ \boldsymbol{W}_i\rangle)$. Because $h_\gamma(\cdot)$ is flat in $(-\infty,\ \min(-1,(k-1)s)]$ and $[\max(k-1,-s),\ +\infty)$, $\langle \boldsymbol{f}^*,\ \boldsymbol{W}_1\rangle$ and $\langle \boldsymbol{f}^*,\ \boldsymbol{W}_i\rangle$ lie in the same interval simultaneously. If $\langle \boldsymbol{f}^*,\ \boldsymbol{W}_1\rangle,\ \langle \boldsymbol{f}^*,\ \boldsymbol{W}_i\rangle \in (-\infty,\ \min(-1,(k-1)s)] < 0$, then all $\langle \boldsymbol{f}^*,\ \boldsymbol{W}_j\rangle < 0$, which is a contradiction to the sum-to-zero property. Thus $\langle \boldsymbol{f}^*,\ \boldsymbol{W}_1\rangle,\ \langle \boldsymbol{f}^*,\ \boldsymbol{W}_i\rangle \in [\max(k-1,-s),\ +\infty)$. If $s < -(k-1)$, then $-s \le \langle \boldsymbol{f}^*,\ \boldsymbol{W}_1\rangle \le k-1$, which is a contradiction. If $0 \ge s \ge -(k-1)$, based on the fact that $\langle \boldsymbol{f}^*,\ \boldsymbol{W}_j\rangle \le k-1$ for all j, then $\langle \boldsymbol{f}^*,\ \boldsymbol{W}_1\rangle = \langle \boldsymbol{f}^*,\ \boldsymbol{W}_i\rangle = k-1$, which contradicts with the assumption. Hence, we must have that $\langle \boldsymbol{f}^*, \boldsymbol{W}_1\rangle \ge \langle \boldsymbol{f}^*, \boldsymbol{W}_j\rangle$ for all $j \neq 1$.

The third step is to show that when $\gamma \le 1/2$, $\langle \boldsymbol{f}^*, \boldsymbol{W}_j\rangle \ge -1$ for all j. Suppose this is not true, and $\langle \boldsymbol{f}^*, \boldsymbol{W}_i\rangle < -1$ for some $i \neq 1$. There must exist $q \in \{1,\dots,k\}$, $q \neq 1, i$, such that $-1 < \langle \boldsymbol{f}^*, \boldsymbol{W}_q\rangle \le k-1$. In this case, we can decrease $\langle \boldsymbol{f}^*, \boldsymbol{W}_q\rangle$ by a small amount and increase $\langle \boldsymbol{f}^*, \boldsymbol{W}_i\rangle$ by the same amount, such that the conditional loss $S(\boldsymbol{x})$ is decreased, which contradicts with the optimality of $\boldsymbol{f}^*$.

The last step is to show that $\langle \boldsymbol{f}^*, \boldsymbol{W}_j\rangle \le 0$ for any $j \neq 1$. If $\langle \boldsymbol{f}^*, \boldsymbol{W}_i\rangle > 0$ for some i, then $\langle \boldsymbol{f}^*, \boldsymbol{W}_1\rangle < k-1$. Otherwise, $\langle \boldsymbol{f}^*, \boldsymbol{W}_1\rangle = k-1$. According to third part, we have $\langle \boldsymbol{f}^*, \boldsymbol{W}_j\rangle = -1,\ j \neq 1$. Especially, $\langle \boldsymbol{f}^*, \boldsymbol{W}_i\rangle = -1 < 0$, which is a contradiction to the assumption $\langle \boldsymbol{f}^*, \boldsymbol{W}_i\rangle > 0$. Then $\langle \boldsymbol{f}^*, \boldsymbol{W}_1\rangle < k-1$. Now we can decrease $\langle \boldsymbol{f}^*, \boldsymbol{W}_i\rangle$ by a small amount, and increase $\langle \boldsymbol{f}^*, \boldsymbol{W}_1\rangle$ by the same amount (Lemma 1), and the conditional loss is decreased. It indicates that the current $\boldsymbol{f}^*$ is not optimal, which is a contradiction. Based on the sum-to-zero property, we show the fact $\langle \boldsymbol{f}^*, \boldsymbol{W}_1\rangle > 0 \geq \langle \boldsymbol{f}^*, \boldsymbol{W}_j\rangle$ for any $j \neq 1$. This completes the first part of the proof, that (3) is Fisher consistent when $\gamma \leq 1/2$ and $s \leq 0$.

We proceed to show the Fisher consistency of (6) when $s \in [-1/(k-1), 0]$. Again, without loss of generality, $P_1 > P_2 \geq \cdots \geq P_k$. By similar arguments as above, one can verify that $\langle \boldsymbol{f}^*, \boldsymbol{W}_1\rangle \geq 0$, and $\langle \boldsymbol{f}^*, \boldsymbol{W}_i\rangle \geq \langle \boldsymbol{f}^*, \boldsymbol{W}_j\rangle$ if $i < j$. Therefore, it remains to show that $\mathbf{0}$ is not the minimizer for the choice of s. To this end, notice that for $s \in [-\frac{1}{k-1}, 0]$, we have $\frac{1}{-s} \geq k-1$. Consequently, there exists $t \in (0, 1]$ such that $\frac{t}{-s} \geq k-1$. Consider a $\boldsymbol{f}$ such that $\langle \boldsymbol{f}, \boldsymbol{W}_1\rangle = \frac{t(k-1)}{k}$ and $\langle \boldsymbol{f}, \boldsymbol{W}_j\rangle = -\frac{t}{k}$ for $j \neq 1$. One can verify that $\boldsymbol{f}$ yields a smaller conditional loss, compared to $\mathbf{0}$. Thus, the robust SVM (6) is Fisher consistent. □

Proof of Theorem 2 We can prove the theorem using a recent technique in the statistical machine learning literature, namely, the Rademacher complexity [3,4,18,19,35,39]. To begin with, let $\sigma = \{\sigma_i;\ i = 1, \ldots, n\}$ be independent and identically distributed random variables, that take 1 and -1 with probability 1/2 each. Denote by S a sample of observations $(\boldsymbol{x}_i, y_i);\ i = 1, \ldots, n$, independent and identically distributed from the underlying distribution $P(\boldsymbol{X}, Y)$. For a function class $\mathscr{F} = \{f : f(\boldsymbol{x}, y)\}$ and given S, we define the empirical Rademacher complexity of $\mathscr{F}$ to be

$$\hat{R}_n(\mathscr{F}) = E_\sigma \left\{ \sup_{f \in \mathscr{F}} \frac{1}{n} \sum_{i=1}^{n} \sigma_i f(\boldsymbol{x}_i, y_i) \right\}.$$

Here E_σ means taking expectation with respect to the distribution of σ. Furthermore, define the Rademacher complexity of $\mathscr{F}$ to be

$$R_n(\mathscr{F}) = E_{\sigma,S} \left\{ \sup_{f \in \mathscr{F}} \frac{1}{n} \sum_{i=1}^{n} \sigma_i f(\boldsymbol{x}_i, y_i) \right\}.$$

Another key step in the proof is to notice that the indicator function in (16) is discontinuous, thus it is difficult to bound the corresponding Rademacher complexity directly. To overcome this challenge, we can consider a continuous upper bound of the indicator function. In particular, for any $\hat{\boldsymbol{f}}$, let I_κ be defined as follows

$$I_\kappa\{\hat{\boldsymbol{f}}(\boldsymbol{x}), y\} = \begin{cases} 1 & \text{if } y \neq \hat{y}_{\hat{\boldsymbol{f}}}(\boldsymbol{x}), \\ 1 - \frac{1}{\kappa} \min_{j \neq y} \langle \hat{\boldsymbol{f}}(\boldsymbol{x}), \boldsymbol{W}_y - \boldsymbol{W}_j\rangle & \text{if } y = \hat{y}_{\hat{\boldsymbol{f}}}(\boldsymbol{x}) \text{ and} \\ & \quad \min_{j \neq y} \langle \hat{\boldsymbol{f}}(\boldsymbol{x}), \boldsymbol{W}_y - \boldsymbol{W}_j\rangle \leq \kappa, \\ 0 & \text{otherwise,} \end{cases}$$

where κ is a small positive number to be determined later. One can verify that I_κ is a continuous upper bound of the indicator function in (16). In the following proof, we focus on bounding the Rademacher complexity of I_κ, where $\hat{\boldsymbol{f}}$ is obtained from the optimization problems (3) and (6).

Our goal is to show that with probability at least $1-\delta$ $(0 < \delta < 1)$, $E[I_\kappa\{\hat{\boldsymbol{f}}(\boldsymbol{X}), Y\}]$ is bounded by the summation of its empirical evaluation, the Rademacher complexity of the function class $\mathscr{F}$, and a penalty term on δ. The proof of Theorem 2 consists of two major steps. In particular, we have the following two lemmas.

Lemma 2 *Let $R_n(\mathscr{F})$ and $\hat{R}_n(\mathscr{F})$ be defined with respect to the I_κ function. Then, with probability at least $1-\delta$,*

$$E[I_\kappa\{\hat{\boldsymbol{f}}(\boldsymbol{X}), Y\}] \le \frac{1}{n}\sum_{i=1}^{n} I_\kappa\{\hat{\boldsymbol{f}}(\boldsymbol{x}_i), y_i\} + 2R_n(\mathscr{F}) + T_n(\delta), \tag{17}$$

where $T_n(\delta) = \{\log(1/\delta)/n\}^{1/2}$.

Moreover, with probability at least $1-\delta$,

$$E[I_\kappa\{\hat{\boldsymbol{f}}(\boldsymbol{X}), Y\}] \le \frac{1}{n}\sum_{i=1}^{n} I_\kappa\{\hat{\boldsymbol{f}}(\boldsymbol{x}_i), y_i\} + 2\hat{R}_n(\mathscr{F}) + 3T_n(\delta/2).$$

Lemma 3 *Let $s = 1/\lambda$. In linear learning, when we use the L_1 penalty, the empirical Rademacher complexity $\hat{R}_n(\mathscr{F}) \le \frac{s}{\kappa}\sqrt{\frac{2\log(2pk-2p)}{n}}$, and when we use the L_2 penalty, $\hat{R}_n(\mathscr{F}) \le \{2(k-1)(ps)^{1/2}\}/(\kappa n^{1/4}) + \{2(ps)^{1/2}\}\Big(\log[e + e\{2p(k-1)\}]/(n^{1/2})\Big)^{1/2}/(\kappa n^{1/4})$. For kernel learning with separable kernel functions, the empirical Rademacher complexity $\hat{R}_n(\mathscr{F}) \le \frac{s(k-1)}{\kappa\sqrt{n}}$.*

Proof of Lemma 2 The proof consists of three parts. For the first part, we use the McDiarmid inequality [34] to bound the left hand side of (17), in terms of its empirical estimation, plus the expectation of their supremum difference, $E(\phi)$, which is to be defined below. For the second part, we show that $E(\phi)$ is bounded by the Rademacher complexity using symmetrization inequalities [42]. For the third part, we prove that one can bound the Rademacher complexity using the empirical Rademacher complexity.

For a given sample S, we define

$$\phi(S) = \sup_{f\in\mathscr{F}}\left(E[I_\kappa\{\hat{\boldsymbol{f}}(\boldsymbol{X}), Y\}] - \frac{1}{n}\sum_{i=1}^{n} I_\kappa\{\hat{\boldsymbol{f}}(\boldsymbol{x}_i), y_i\}\right).$$

Let $S^{(i,\boldsymbol{x})} = \{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}'_i, y_i), \ldots, (\boldsymbol{x}_n, y_n)\}$ be another sample from $P(\boldsymbol{X}, Y)$, where the difference between S and $S^{(i,\boldsymbol{x})}$ is only on the $\boldsymbol{x}$ value of their ith pair. By definition, we have

$$\begin{aligned}|\phi(S) - \phi(S^{(i,\boldsymbol{x})})| = \Bigg|&\sup_{f\in\mathscr{F}}\left(E[I_\kappa\{\hat{\boldsymbol{f}}(\boldsymbol{X}), Y\}] - \frac{1}{n}\sum_{S} I_\kappa\{\hat{\boldsymbol{f}}(\boldsymbol{x}_i), y_i\}\right)\\ &- \sup_{f\in\mathscr{F}}\left(E[I_\kappa\{\hat{\boldsymbol{f}}(\boldsymbol{X}), Y\}] - \frac{1}{n}\sum_{S^{(i,\boldsymbol{x})}} I_\kappa\{\hat{\boldsymbol{f}}(\boldsymbol{x}_i), y_i\}\right)\Bigg|.\end{aligned}$$

For simplicity, suppose that $\boldsymbol{f}^S$ is the function that achieves the supremum of $\phi(S)$. We note that the case of no function achieving the supremum can be treated analogously, with only additional discussions on the arbitrarily small difference between $\phi(\boldsymbol{f})$ and its supremum. Thus, we omit the details here. We have that,

$$\begin{aligned}|\phi(S)-\phi(S^{(i,\boldsymbol{x})})| \leq &\Bigg| E[I_\kappa\{\boldsymbol{f}^S(\boldsymbol{X}),Y\}]-\frac{1}{n}\sum_{S} I_\kappa\{\boldsymbol{f}^S(\boldsymbol{x}_i),y_i\} \\ &- E[I_\kappa\{\boldsymbol{f}^S(\boldsymbol{X}),Y\}]+\frac{1}{n}\sum_{S^{(i,\boldsymbol{x})}} I_\kappa\{\boldsymbol{f}^S(\boldsymbol{x}_i),y_i\}\Bigg|. \\ &= \frac{1}{n}\left|\sum_{S} I_\kappa\{\boldsymbol{f}^S(\boldsymbol{x}_i),y_i\}-\sum_{S^{(i,\boldsymbol{x})}} I_\kappa\{\boldsymbol{f}^S(\boldsymbol{x}_i),y_i\}\right| \\ &\leq \frac{1}{n}.\end{aligned}$$

Next, by the McDiarmid inequality, we have that for any $t>0$, $\mathrm{pr}[\phi(S)-E\{\phi(S)\}\geq t]\leq \exp[-(2t^2)/\{2n(1/n)^2\}]$, or equivalently, with probability at least $1-\delta$, $\phi(S)-E\{\phi(S)\}\leq T_n(\delta)$. Consequently, we have that with probability at least $1-\delta$, $E[I_\kappa\{\boldsymbol{f}^S(\boldsymbol{X}),Y\}]\leq n^{-1}\sum_{i=1}^n I_\kappa\{\hat{\boldsymbol{f}}(\boldsymbol{x}_i),y_i\}+E\{\phi(S)\}+T_n(\delta)$. This completes the first part of the proof.

For the second part, we bound $E\{\phi(S)\}$ by the corresponding Rademacher complexity. To this end, define $S'=\{(\boldsymbol{x}_i',y_i');\ i=1,\ldots,n\}$ as an independent duplicate sample of size n with the identical distribution as S. Denote by E_S the action of taking expectation with respect to the distribution of S, and define $E_{S'}$ analogously. By definition, we have that $E_{S'}\big[n^{-1}\sum_{S'} I_\kappa\{\hat{\boldsymbol{f}}(\boldsymbol{x}_i'),y_i'\}\mid S\big]=E[I_\kappa\{\hat{\boldsymbol{f}}(\boldsymbol{X}),Y\}]$, and $E_{S'}\big[n^{-1}\sum_S I_\kappa\{\hat{\boldsymbol{f}}(\boldsymbol{x}_i),y_i\}\mid S\big]=n^{-1}\sum_S I_\kappa\{\hat{\boldsymbol{f}}(\boldsymbol{x}_i),y_i\}$. Then, by Jensen's inequality and the property of σ, we have that

$$\begin{aligned}E\{\phi(S)\} &= E_S\left(\sup_{f\in\mathscr{F}} E_{S'}\left[\frac{1}{n}\sum_{S'} I_\kappa\{\hat{\boldsymbol{f}}(\boldsymbol{x}_i'),y_i'\}-\frac{1}{n}\sum_{S} I_\kappa\{\hat{\boldsymbol{f}}(\boldsymbol{x}_i),y_i\}\right]\mid S\right) \\ &\leq E_{S,S'}\left[\sup_{f\in\mathscr{F}}\frac{1}{n}\sum_{S'} I_\kappa\{\hat{\boldsymbol{f}}(\boldsymbol{x}_i'),y_i'\}-\frac{1}{n}\sum_{S} I_\kappa\{\hat{\boldsymbol{f}}(\boldsymbol{x}_i),y_i\}\right] \\ &= E_{S,S',\sigma}\big[\sup_{f\in\mathscr{F}}\frac{1}{n}\sum_{S'}\sigma_i I_\kappa\{\hat{\boldsymbol{f}}(\boldsymbol{x}_i'),y_i'\}-\frac{1}{n}\sum_{S}\sigma_i I_\kappa\{\hat{\boldsymbol{f}}(\boldsymbol{x}_i),y_i\}\big] \\ &\leq 2R_n\{\mathscr{F}(p,k,s)\}.\end{aligned}$$

Hence the second part is proved.

In the third step, we need to bound $R_n(\mathscr{F})$ using $\hat{R}_n(\mathscr{F})$. This step is analogous to the first part, and we omit the details here. Briefly speaking, one can apply the McDiarmid inequality on $\hat{R}_n(\mathscr{F})$ and the corresponding expectation $R_n(\mathscr{F})$. Similar to the first part of this proof, we can show that with probability at least $1-\delta$, $R_n(\mathscr{F}) \le \hat{R}_n(\mathscr{F}) + 2T_n(\delta)$.

The final results of Lemma 2 can be obtained by choosing the confidence $1-\delta/2$ in the first and third steps, and combining the inequalities of the three steps. □

Proof of Lemma 3 First, we prove that for the obtained $\hat{\boldsymbol{f}}$, $J(\hat{\boldsymbol{f}}) \le s$. To see this, notice that for $\boldsymbol{\beta}_j = 0$ and $\beta_{j,0} = 0$, we have

$$\frac{1}{n}\sum_{i=1}^{n} I_\kappa\{\boldsymbol{f}(\boldsymbol{x}_i), y_i\} \le 1.$$

On the other hand, $\hat{\boldsymbol{f}}$ is the solution to the optimization problems in (3) or (6), hence

$$\lambda J(\hat{\boldsymbol{f}}) \le \frac{1}{n}\sum_{i=1}^{n} I_\kappa\{\hat{\boldsymbol{f}}(\boldsymbol{x}_i), y_i\} + \lambda J(\hat{\boldsymbol{f}}) \le \frac{1}{n}\sum_{i=1}^{n} I_\kappa\{\boldsymbol{f}(\boldsymbol{x}_i), y_i\},$$

which yields $J(\hat{\boldsymbol{f}}) \le s$.

For the L_1 penalized learning, one can bound the corresponding Rademacher complexity $\hat{R}_n(\mathscr{F})$ in the following way. In particular, by Lemma 4.2 in [35], we have that $\hat{R}_n(\mathscr{F})$ is upper bounded by

$$\frac{1}{\kappa}\hat{R}_n^*(\mathscr{F}) = \frac{1}{\kappa}E_\sigma\left\{\sup_{\sum_{j=1}^{k-1}\|\boldsymbol{\beta}_j\|_1 < s} \frac{1}{n}\sum_{i=1}^{n}\sigma_i\left\{\sum_{j=1}^{k-1}\boldsymbol{x}_i^{\mathrm{T}}\boldsymbol{\beta}_j\right\}\right\}, \tag{18}$$

because the continuous indicator function is Lipschitz with constant $1/\kappa$, and elements in $\boldsymbol{W}_j$ are bounded by 1. Without loss of generality, we can rewrite (18) as the following

$$\frac{1}{\kappa}\hat{R}_n^*\{\mathscr{F}(p,k,s)\} = \frac{1}{\kappa}E_\sigma\left\{\sup_{\|\boldsymbol{\gamma}\|_1 < s}\frac{1}{n}\sum_{i=1}^{n}\sigma_i\boldsymbol{\gamma}^{\mathrm{T}}\boldsymbol{x}_i^*\right\},$$

where $\boldsymbol{\gamma}$ can be treated as a vector that contains all the elements in $\boldsymbol{\beta}_j$ for $j = 1,\ldots,k-1$, and $\boldsymbol{x}_i^*$ is defined accordingly. Next, by Theorem 10.10 in [35], we have that $\hat{R}_n^*\{\mathscr{F}(p,k,s)\} \le s\sqrt{\frac{2\log(2pk-2p)}{n}}$. Thus, $\hat{R}_n\{\mathscr{F}(p,k,s)\} \le \frac{s}{\kappa}\sqrt{\frac{2\log(2pk-2p)}{n}}$ for L_1 penalized linear learning.

For L_2 penalized learning, the proof is analogous to that of Lemma 8 in [49], and we omit the details here.

For kernel learning, notice that one can include the intercept in the original predictor space (i.e., augment $\boldsymbol{x}$ to include a constant 1 before the other predictors), and define a new kernel function accordingly. This new kernel is also positive definite and separable with a bounded kernel function. By Mercer's Theorem, this introduces a new RKHS $\mathcal{H}$. Next, by a similar argument as for (18), we have that the original Rademacher complexity is upper bounded by

$$
\begin{aligned}
\frac{1}{\kappa}\hat{R}_n^*(\mathscr{F}) &= \frac{1}{\kappa}E_\sigma\left[\sup_{\sum_j \|f_j\|_{\mathcal{H}}^2 \le s} \frac{1}{n}\sum_{i=1}^n \sigma_i \left\{\sum_{j=1}^{k-1} f_j(\boldsymbol{x}_i)\right\}\right], \\
&\le \frac{k-1}{\kappa}E_\sigma\left[\sup_{\|f\|_{\mathcal{H}}^2 \le s} \frac{1}{n}\sum_{i=1}^n \sigma_i f(\boldsymbol{x}_i)\right], \\
&\le \frac{k-1}{\kappa}\frac{s}{\sqrt{n}},
\end{aligned}
$$

where the last inequality follows from Theorem 5.5 in [35]. Hence, we have that for kernel learning, $\hat{R}_n(\mathscr{F}) \le \frac{s(k-1)}{\kappa\sqrt{n}}$. □

The proof of Theorem 2 is thus finished by combining Lemmas 2 and 3, and the fact that the continuous indicator function I_κ is an upper bound of the indicator function for any κ. □

References

1. Arora, S., Bhattacharjee, D., Nasipuri, M., Malik, L., Kundu, M., Basu, D.K.: Performance Comparison of SVM and ANN for Handwritten Devnagari Character Recognition. arXiv preprint arXiv:1006.5902 (2010)
2. Bache, K., Lichman, M.: UCI Machine Learning Repository. University of California, Irvine, School of Information and Computer Sciences. http://archive.ics.uci.edu/ml (2013)
3. Bartlett, P.L., Mendelson, S.: Rademacher and Gaussian complexities: risk bounds and structural results. J. Mach. Learn. Res. **3**, 463–482 (2002)
4. Bartlett, P.L., Bousquet, O., Mendelson, S.: Local rademacher complexities. Ann. Stat. **33**(4), 1497–1537 (2005)
5. Bartlett, P.L., Jordan, M.I., McAuliffe, J.D.: Convexity, classification, and risk bounds. J. Am. Stat. Assoc. **101**, 138–156 (2006)
6. Boser, B.E., Guyon, I.M., Vapnik, V.N.: A training algorithm for optimal margin classifiers. In: Haussler, D. (ed.) Proceedings of the Fifth Annual Workshop on Computational Learning Theory, COLT '92, pp. 144–152. Association for Computing Machinery, New York (1992). https://doi.org/10.1145/130385.130401
7. Caruana, R., Karampatziakis, N., Yessenalina, A.: An empirical evaluation of supervised learning in high dimensions. In: Proceedings of the 25th International Conference on Machine Learning, pp. 96–103. ACM (2008)
8. Cortes, C., Vapnik, V.N.: Support vector networks. Mach. Learn. **20**, 273–297 (1995)
9. Crammer, K., Singer, Y.: On the algorithmic implementation of multiclass kernel-based vector machines. J. Mach. Learn. Res. **2**, 265–292 (2001)
10. Cristianini, N., Shawe-Taylor, J.S.: An Introduction to Support Vector Machines, 1st edn. Cambridge University Press, Cambridge (2000)
11. Demšar, J., Curk, T., Erjavec, A., Črt Gorup, Hočevar, T., Milutinovič, M., Možina, M., Polajnar, M., Toplak, M., Starič, A., Štajdohar, M., Umek, L., Žagar, L., Žbontar, J., Žitnik, M., Zupan, B.: Orange: data mining toolbox in python. J. Mach. Learn. Res. 14:2349–2353. http://jmlr.org/papers/v14/demsar13a.html (2013)
12. Freund, Y., Schapire, R.E.: A Desicion-theoretic generalization of on-line learning and an application to boosting. J. Comput. Syst. Sci. **55**(1), 119–139 (1997)
13. Guermeur, Y., Monfrini, E.: A quadratic loss multi-class SVM for which a radius-margin bound applies. Informatica **22**(1), 73–96 (2011)
14. Hastie, T.J., Tibshirani, R.J., Friedman, J.H.: The Elements of Statistical Learning, 2nd edn. Springer, New York (2009)

15. Hsieh, C., Chang, K., Lin, C., Keerthi, S., Sundarajan, S.: A dual coordinate descent method for large-scale linear SVM. In: Proceedings of the 25th International Conference on Machine Learning, Proceeding ICML '08, pp. 408–415 (2008)
16. Justino, E.J.R., Bortolozzi, F., Sabourin, R.: A comparison of SVM and HMM classifiers in the off-line signature verification. Pattern Recognit. Lett. **26**(9), 1377–1385 (2005)
17. Kiwiel, K., Rosa, C., Ruszczynski, A.: Proximal decomposition via alternating linearization. SIAM J. Optim. **9**(3), 668–689 (1999)
18. Koltchinskii, V.: Local Rademacher complexities and oracle inequalities in risk minimization. Ann. Stat. **34**(6), 2593–2656 (2006)
19. Koltchinskii, V., Panchenko, D.: Empirical margin distributions and bounding the generalization error of combined classifiers. Ann. Stat. **30**(1), 1–50 (2002)
20. Le Thi, H.A., Pham Dinh, T.: Solving a class of linearly constrained indefinite quadratic problems by DC algorithms. J. Glob. Optim. **11**(3), 253–285 (1997)
21. Le Thi, H.A., Pham Dinh, T.: The DC (difference of convex functions) programming and DCA revisited with dc models of real world nonconvex optimization problems. Ann. Oper. Res. **133**, 23–46 (2005)
22. Le Thi, H.A., Pham Dinh, T.: The State of the Art in DC Programming and DCA. Research Report (60 pages), Lorraine University (2013)
23. Le Thi, H.A., Pham Dinh, T.: Recent advances in DC programming and DCA. Trans. Comput. Collect. Intell. **8342**, 1–37 (2014)
24. Le Thi, H.A., Le, H.M., Pham Dinh, T.: A dc programming approach for feature selection in support vector machines learning. Adv. Data Anal. Classif. **2**(3), 259–278 (2008)
25. Le Thi, H.A., Huynh, V.N., Pham Dinh, T.: DC programming and DCA for general DC programs. Adv. Intell. Syst. Comput. 15–35. ISBN 978-3-319-06568-7 (2014)
26. Lee, Y., Lin, Y., Wahba, G.: Multicategory support vector machines, theory, and application to the classification of microarray data and satellite radiance data. J. Am. Stat. Assoc. **99**, 67–81 (2004)
27. Lin, X., Wahba, G., Xiang, D., Gao, F., Klein, R., Klein, B.: Smoothing spline ANOVA models for large data sets with bernoulli observations and the randomized GACV. Ann. Stat. **28**(6), 1570–1600 (2000)
28. Lin, X., Pham, M., Ruszczynski, A.: Alternating linearization for structured regularization problem. J. Mach. Learn. Res. **15**, 3447–3481 (2014)
29. Lin, Y.: Some Asymptotic Properties of the Support Vector Machine. Technical Report 1044r, Department of Statistics, University of Wisconsin, Madison (1999)
30. Liu Y (2007) Fisher consistency of multicategory support vector machines. In: Eleventh International Conference on Artificial Intelligence and Statistics, pp. 289–296
31. Liu, Y., Shen, X.: Multicategory ψ-learning. J. Am. Stat. Assoc. **101**, 500–509 (2006)
32. Liu, Y., Yuan, M.: Reinforced multicategory support vector machines. J. Comput. Gr. Stat. **20**(4), 901–919 (2011)
33. Liu, Y., Zhang, H.H., Wu, Y.: Soft or hard classification? Large margin unified machines. J. Am. Stat. Assoc. **106**, 166–177 (2011)
34. McDiarmid, C.: On the method of bounded differences. In: Surveys in Combinatorics, Cambridge University Press, Cambridge, pp. 148–188 (1989)
35. Mohri, M., Rostamizadeh, A., Talwalkar, A.: Foundations of Machine Learning. MIT Press, Cambridge, MA (2012)
36. Nesterov, Y.: Efficiency of coordinate descent methods on huge-scale optimization problems. SIAM J. Optim. **22**(4), 341–362 (2012)
37. Pang, J.S., Razaviyayn, M., Alvarado, A.: Computing B-stationary points of nonsmooth DC programs. Math. Oper. Res. **42**, 95–118 (2016)
38. Platt, J.: Fast training of support vector machines using sequential minimal optimization. In: Schölkopf, B., Burges, J.C., Smola, A.J. (eds.) Advances in Kernel Methods: Support Vector Learning, pp. 185–208. MIT Press, Cambridge, MA, USA (1999)
39. Shawe-Taylor, J.S., Cristianini, N.: Kernel Methods for Pattern Analysis, 1st edn. Cambridge University Press, Cambridge (2004)
40. Steinwart, I., Scovel, C.: Fast rates for support vector machines using Gaussian kernels. Ann. Stat. **35**(2), 575–607 (2007)
41. Tseng, P.: A coordinate gradient descent method for linearly constrained smooth optimization and support vector machines training. J. Comput. Optim. Appl. **47**(4), 179–206 (2010)

42. van der Vaart, A.W., Wellner, J.A.: Weak Convergence and Empirical Processes with Application to Statistics, 1st edn. Springer, Berlin, New York, NY (2000)
43. Vapnik, V.N.: Statistical Learning Theory. Wiley, New York (1998)
44. Wahba, G.: Support vector machines, reproducing kernel Hilbert spaces and the randomized GACV. In: Schölkopf, B., Burges, J.C., Smola, A.J. (eds.) Advances in Kernel Methods: Support Vector learning, pp. 69–88. MIT Press, Cambridge, MA, USA (1999)
45. Wang, L., Shen, X.: On L_1-norm multi-class support vector machines: methodology and theory. J. Am. Stat. Assoc. **102**, 595–602 (2007)
46. Wang, L., Zhu, J., Zou, H.: The doubly regularized support vector machine. Stat. Sin. **16**, 589–615 (2006)
47. Wu, Y., Liu, Y.: On multicategory truncated-hinge-loss support vector. In: Prediction and Discovery: AMS-IMS-SIAM Joint Summer Research Conference, Machine and Statistical Learning: Prediction and Discovery, June 25–29, 2006, Snowbird, Utah, American Mathematical Society, vol. 443, pp. 49–58 (2006)
48. Wu, Y., Liu, Y.: Robust truncated hinge loss support vector machines. J. Am. Stat. Assoc. **102**(479), 974–983 (2007)
49. Zhang, C., Liu, Y.: Multicategory angle-based large-margin classification. Biometrika **101**(3), 625–640 (2014)
50. Zhang, C., Liu, Y., Wang, J., Zhu, H.: Reinforced angle-based multicategory support vector machines. J. Comput. Gr. Stat. **25**, 806–825 (2016)