Optimal Margin Distribution Machine

Teng Zhang and Zhi-Hua Zhou, *Fellow, IEEE*

Abstract—Support Vector Machine (SVM) has always been one of the most successful learning algorithms, with the central idea of maximizing the *minimum margin*, i.e., the smallest distance from the instances to the classification boundary. However, recent theoretical results disclosed that maximizing the minimum margin does not necessarily lead to better generalization performance, and instead, the *margin distribution* has been proven to be more crucial. Based on this idea, we propose the Optimal margin Distribution Machine (ODM), which can achieve a better generalization performance by optimizing the margin distribution explicitly. We characterize the margin distribution by the first- and second-order statistics, i.e., the margin mean and variance. The proposed method is a general learning approach which can be applied in any place where SVMs are used, and its superiority is verified both theoretically and empirically in this paper.

Index Terms—margin, margin distribution, minimum margin, classification.

1 Introduction

Support Vector Machine (SVM) [1] has always been one of the most successful learning algorithms. The idea is to identify a classification boundary having a large margin for all the training instances, and the resultant optimization can be accomplished by a quadratic programming (QP) problem. Although SVMs have a long history of literatures, there are still great efforts on improving SVMs [2], [3], [4].

It is well known that SVMs can be viewed as a learning approach trying to maximize the minimum margin of training instances, i.e., the smallest distance from the instances to the classification boundary, and the margin theory [5] provided a good support to the generalization performance of SVMs. It is noteworthy that there is also a long history of applying margin theory to explain the good generalization of AdaBoost [6], a major representative of ensemble methods [7]. Specifically, [8] first suggested margin theory to explain the phenomenon that AdaBoost seems resistant to over-fitting; soon after, [9] indicated that the minimum margin is crucial and developed a boosting-style algorithm, named Arc-gv, which is able to maximize the minimum margin but with a poor generalization performance. Later, [10] observed that although Arc-gv produced a larger minimum margin, its margin distribution is quite poor. Nowadays, it is well-accepted that the margin distribution is more crucial to the generalization performance of AdaBoost, and relevant generalization error bounds have also been proven in [11]. All these theoretical studies, however, focused on boosting-style algorithms, whereas the influence of the margin distribution for SVMs in practice has not been well exploited. In addition, the central idea of SVMs is to maximize the minimum margin, so there may still exist large space to further enhance for SVMs. Finally, SVMs are widely used learning methods in practice, so a remarkable improvement of SVMs will play a big role in many real applications.

• *T. Zhang and Z.-H. Zhou are with National Key Laboratory for Novel Software Technology, Nanjing University, Nanjing 210023, China. E-mail: {zhangt, zhouzh}@lamda.nju.edu.cn*

In this paper, we propose the Optimal Margin Distribution Machine (ODM), whose idea is to optimize the margin distribution so as to get better generalization performance. Inspired by the theoretical work [11], we characterize the margin distribution by the first- and second-order statistics, that is, maximizing the margin mean and minimizing the margin variance simultaneously. For optimization, we propose a dual coordinate descent method for kernel ODM, and a SVRG method for large scale linear kernel ODM. We also extend the block coordinate descent method for multi-class kernel ODM (mcODM). Extensive experiments on fifty four binary data sets and twenty two multi-class data sets show the superiority of our methods to SVMs in comparisons with other state-of-the-art methods, verifying that the margin distribution is more crucial than minimum margin for SVM-style learning approaches.

A preliminary version of ODM appeared in a conference paper [12]. Compared with the original version, we propose a new approach which can characterize the margin distribution adaptively by a well-designed loss function, and is more efficient for nonlinear kernels owing to the avoidance of computation of matrix inverse. Based on Gaussian complexity, we provide a better generalization error bound rather than the original expected error bound. We call this new approach as Optimal margin Distribution Machine (ODM), whereas the preliminary method was called Large margin Distribution Machine (LDM) and denoted by ODM^L for convenience. Up to now, there are a lot of works based on ODM^L, just to name a few, [13] extends it to semi-supervised learning, an important case of weakly supervised learning [14] and handle unequal misclassification cost. [15] proposes a margin distribution machine for clustering. [16] applies it to feature elimination.

The rest of this paper is organized as follows. Section 2 introduces some preliminaries. Section 3 and 4 present the formulation and optimization of ODM and mcODM, respectively. Section 5 presents the theoretical analysis. Section 6 discusses about some related works. Section 7 gives extensive empirical studies, and finally Section 8 concludes with future work.

2 PRELIMINARIES

We denote by $\mathcal{X} \subseteq \mathbb{R}^d$ the instance space and $\mathcal{Y} = [k]$ the label set, where $k \geq 2$ and $[k] = \{1, \ldots, k\}$. Let $\mathcal{D}$ be an unknown (underlying) distribution over $\mathcal{X} \times \mathcal{Y}$. A training set $\mathcal{S} = \{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_m, y_m)\} \in (\mathcal{X} \times \mathcal{Y})^m$ is drawn identically and independently (i.i.d.) according to the distribution $\mathcal{D}$. Let $\phi : \mathcal{X} \mapsto \mathbb{H}$ be a feature mapping associated to some positive definite kernel κ. The hypothesis is defined based on k weight vectors $\boldsymbol{w}_1, \ldots, \boldsymbol{w}_k \in \mathbb{H}$, where each one defines a scoring function $\boldsymbol{x} \mapsto \boldsymbol{w}_l^\top \phi(\boldsymbol{x}), l \in [k]$. The label of instance $\boldsymbol{x}$ is the one resulting in the largest score, i.e., $h(\boldsymbol{x}) = \operatorname{argmax}_{l \in [k]} \boldsymbol{w}_l^\top \phi(\boldsymbol{x})$. This decision function naturally leads to the following definition of margin for a labeled instance $(\boldsymbol{x}, y)$:

$$\gamma_h(\boldsymbol{x}, y) = \boldsymbol{w}_y^\top \phi(\boldsymbol{x}) - \max_{l \neq y} \boldsymbol{w}_l^\top \phi(\boldsymbol{x}), \tag{1}$$

thus h misclassifies $(\boldsymbol{x}, y)$ iff it produces a negative margin for this instance.

2.1 Binary SVMs

For binary classification, Eq. (1) can be equivalently rewritten as $\gamma_h(\boldsymbol{x}, y) = (1_{y=1} - 1_{y=2})(\boldsymbol{w}_1 - \boldsymbol{w}_2)^\top \phi(\boldsymbol{x})$, where $1_{(\cdot)}$ is the indicator function that returns 1 when the argument holds, and 0 otherwise. To simplify the expression, we change the label set to $\{1, -1\}$, then we have the specific definition of margin for binary classification: $\gamma_h(\boldsymbol{x}, y) = (1_{y=1} - 1_{y=-1})(\boldsymbol{w}_1 - \boldsymbol{w}_{-1})^\top \phi(\boldsymbol{x}) = y\boldsymbol{w}^\top \phi(\boldsymbol{x})$, where $\boldsymbol{w} = \boldsymbol{w}_1 - \boldsymbol{w}_{-1}$.

It is easy to find that all $\gamma_h(\boldsymbol{x}_i, y_i)$ will be positive for separable data, i.e., the training instances can be separated without error. By scaling it with $1/\|\boldsymbol{w}\|$, we can get the signed geometric distance from $(\phi(\boldsymbol{x}_i), y_i)$ to the hyperplane $\boldsymbol{w}^\top \phi(\boldsymbol{x}) = 0$, i.e., $\hat{\gamma}_h(\boldsymbol{x}_i, y_i) = y_i \boldsymbol{w}^\top \phi(\boldsymbol{x}_i)/\|\boldsymbol{w}\| = \gamma_h(\boldsymbol{x}_i, y_i)/\|\boldsymbol{w}\|$. The hard-margin SVM is regarded as the maximization of the minimum distance,

$$\max_{\boldsymbol{w}} \ \gamma/\|\boldsymbol{w}\|, \text{ s.t. } y_i \boldsymbol{w}^\top \phi(\boldsymbol{x}_i) \geq \gamma, \ \forall i.$$

Since the value of γ doesn't have influence on the optimization, we can simply set it as 1. Note that maximizing $1/\|\boldsymbol{w}\|$ is equivalent to minimizing $\|\boldsymbol{w}\|^2/2$, we can get the classic formulation of hard-margin SVM as follows:

$$\min_{\boldsymbol{w}} \ \frac{1}{2}\|\boldsymbol{w}\|^2, \text{ s.t. } y_i \boldsymbol{w}^\top \phi(\boldsymbol{x}_i) \geq 1, \ \forall i.$$

For non-separable data, i.e., the training instances can't be separated without error, soft-margin SVM is posed,

$$\begin{aligned} &\min_{\boldsymbol{w}, \xi_i} \ \frac{1}{2}\|\boldsymbol{w}\|^2 + \frac{\lambda}{m}\sum_{i=1}^m \xi_i, \\ &\text{s.t. } y_i \boldsymbol{w}^\top \phi(\boldsymbol{x}_i) \geq 1 - \xi_i, \ \xi_i \geq 0, \ \forall i. \end{aligned} \tag{2}$$

where $\boldsymbol{\xi}^\top = [\xi_1, \ldots, \xi_m]$ are slack variables which measure the losses of instances, and λ is a trading-off parameter. Again, note that minimizing $\|\boldsymbol{w}\|^2/2$ and $\sum_{i=1}^m \xi_i/m$ is equivalent to maximizing $1/\|\boldsymbol{w}\|$ and $1 - \sum_{i=1}^m \xi_i/m$ respectively, so there exists a constant $\bar{\lambda}$ such that Eq. (2) can be equivalently reformulated as,

$$\begin{aligned} &\max_{\boldsymbol{w}, \xi_i} \ \frac{1}{\|\boldsymbol{w}\|}\left(1 - \frac{\bar{\lambda}}{m}\sum_{i=1}^m \xi_i\right), \\ &\text{s.t. } y_i \boldsymbol{w}^\top \phi(\boldsymbol{x}_i) \geq 1 - \xi_i, \ \xi_i \geq 0, \ \forall i. \end{aligned} \tag{3}$$

Denote $\hat{\gamma}_0 = 1/\|\boldsymbol{w}\|$ and $\hat{\xi}_i = \xi_i/\|\boldsymbol{w}\|$, then we have

$$\frac{1}{\|\boldsymbol{w}\|}\left(1 - \frac{\bar{\lambda}}{m}\sum_{i=1}^m \xi_i\right) = \hat{\gamma}_0 - \frac{\bar{\lambda}}{m}\sum_{i=1}^m \hat{\xi}_i,$$

and

$$\begin{aligned} &y_i \boldsymbol{w}^\top \phi(\boldsymbol{x}_i) \geq 1 - \xi_i \iff y_i \boldsymbol{w}^\top \phi(\boldsymbol{x}_i) \geq \hat{\gamma}_0 \|\boldsymbol{w}\| - \xi_i \\ &\iff \frac{y_i \boldsymbol{w}^\top \phi(\boldsymbol{x}_i)}{\|\boldsymbol{w}\|} \geq \hat{\gamma}_0 - \frac{\xi_i}{\|\boldsymbol{w}\|} \iff \hat{\gamma}_h(\boldsymbol{x}_i, y_i) \geq \hat{\gamma}_0 - \hat{\xi}_i. \end{aligned}$$

Since $\xi_i \geq 0$ iff $\hat{\xi}_i \geq 0$, Eq. (3) can be rewritten as

$$\max_{\hat{\gamma}_0, \hat{\xi}_i} \ \hat{\gamma}_0 - \frac{\bar{\lambda}}{m}\sum_{i=1}^m \hat{\xi}_i, \text{ s.t. } \hat{\gamma}_h(\boldsymbol{x}_i, y_i) \geq \hat{\gamma}_0 - \hat{\xi}_i, \ \hat{\xi}_i \geq 0, \ \forall i.$$

Note that $\hat{\gamma}_0$ indeed characterizes the top-p minimum margin, i.e., the p-th smallest value in $\{y_i \boldsymbol{w}^\top \phi(\boldsymbol{x}_i), i \in [m]\}$; hence SVMs consider only a single-point margin and have not exploited the whole margin distribution, which may be misleading in some situations. See Figure 1 for an illustration [17].

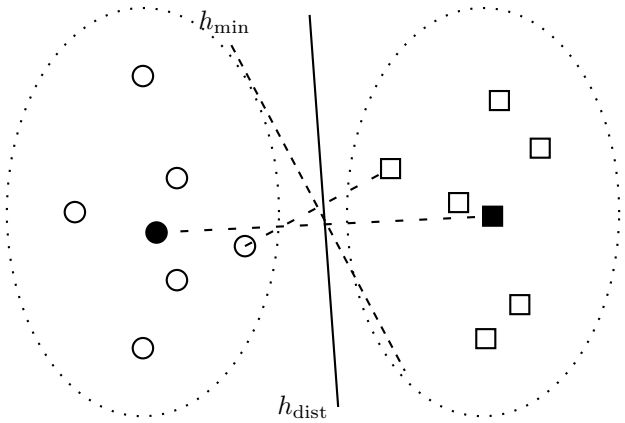

Fig. 1. A simple illustration of linear separators optimizing the minimum margin and margin distribution, respectively. Dotted ellipses are two underlying distributions, from which circles and squares are instances sampled. Solid circle and square are mean instances (not necessarily real instances in the training data). $h_{\min}$ and h_{dist} are decision hyperplanes achieved by optimizing the minimum margin and margin distribution, respectively.

2.2 Multi-Class SVMs

For multi-class classification, there are roughly four types of SVMs can be applied:

1) One-vs-rest SVM (ovrSVM) consists of learning k scoring functions $h_l : \mathcal{X} \mapsto \{-1, +1\}$, $l \in [k]$, each seeking to discriminate one class l from all the others, as can be seen it need train k SVM models.
2) One-vs-one SVM (ovoSVM) [18] determines the scoring functions for all the combinations of class pairs, so it need train $k(k-1)/2$ SVM models.
3) Error-correcting output code SVM (ecocSVM) [19] is a generalization of the former two methods. This technique assigns to each class $l \in [k]$ a code word with length c, which serves as a signature for this class. After training c binary SVM models $h_1(\cdot), \ldots, h_c(\cdot)$, the class predicted for each testing instance is the one

whose signatures is the closest to $[h_1(\boldsymbol{x}), \ldots, h_c(\boldsymbol{x})]$ in Hamming distance.

4) Multi-class SVM (mcSVM) is extended from the binary SVM with joint formulation. Specifically, [20] directly optimizes the minimum margin defined in Eq. (1). CappedSVM [21] optimized the truncated loss to handle outliers. A new multi-class SVM was developed in [22] which considers the margin for each class pair.

These four methods can be classified into two groups. The first group includes the first three methods by converting the multi-class classification problem into a set of binary classification problems. The weakness of these methods is that they may produce unclassifiable regions and the computational costs are very large in practice since a lot of hyperparamers need to be tuned. On the other hand, mcSVM belongs to the second group. It directly determines all the scoring functions at once, so the time cost is usually less than the former methods. In addition, the unclassifiable regions are also resolved. Note that all these methods are besed on binary SVM. So all these methods have the same weakness with binary SVM, i.e., they consider only a single-point margin and have not exploited the whole margin distribution.

3 ODM

The two most straightforward statistics for characterizing margin distribution are the first- and second-order statistics, that is, margin mean and variance. In this section, we introduce ODM^L and ODM, which both are formulated as a QP with decoupled box constraints, so we propose a dual coordinate descent method ODM$_{dcd}$ to solve them uniformly. We also extend a SVRG based method ODM$_{svrg}$ for large scale problems.

3.1 ODM^L

Denote $\boldsymbol{X}$ as the matrix whose i-th column is $\phi(\boldsymbol{x}_i)$, i.e., $\boldsymbol{X} = [\phi(\boldsymbol{x}_1), \ldots, \phi(\boldsymbol{x}_m)]$, $\boldsymbol{y}^\top = [y_1, \ldots, y_m]$, and $\boldsymbol{Y}$ is a $m \times m$ diagonal matrix with $y_1, \ldots, y_m$ as the diagonal elements. According to the definition, the margin mean and margin variance are

$$\gamma_m = \frac{1}{m}\sum_{i=1}^{m} y_i \boldsymbol{w}^\top \phi(\boldsymbol{x}_i) = \frac{1}{m}(\boldsymbol{X}\boldsymbol{y})^\top \boldsymbol{w},$$
$$\gamma_v = \frac{1}{m}\sum_{i=1}^{m} (y_i \boldsymbol{w}^\top \phi(\boldsymbol{x}_i) - \bar{\gamma})^2 = \boldsymbol{w}^\top \boldsymbol{X} \frac{m\boldsymbol{I} - \boldsymbol{y}\boldsymbol{y}^\top}{m^2} \boldsymbol{X}^\top \boldsymbol{w}.$$

Inspired by the theoretical work [11], we attempt to maximize the margin mean and minimize the margin variance simultaneously. The most natural idea is to add the margin mean γ_m and the margin variance γ_v to the objective function of SVM explicitly, which leads to

$$\begin{aligned} \min_{\boldsymbol{w},\xi_i} \ & \frac{1}{2}\|\boldsymbol{w}\|^2 + \mu_1\gamma_v - \mu_2\gamma_m + \frac{\lambda}{m}\sum_{i=1}^{m}\xi_i, \\ \text{s.t. } & y_i \boldsymbol{w}^\top \phi(\boldsymbol{x}_i) \geq 1 - \xi_i,\ \xi_i \geq 0,\ \forall i. \end{aligned} \tag{4}$$

where μ_1 and μ_2 are trading-off parameters. It's evident that ODM^L subsumes SVM when μ_1 and μ_2 equal 0.

Substituting margin mean and margin variance, Eq. (4) leads to the following quadratic programming problem,

$$\begin{aligned} \min_{\boldsymbol{w},\boldsymbol{\xi}} \ & \frac{1}{2}\|\boldsymbol{w}\|^2 + \boldsymbol{w}^\top \boldsymbol{X} \frac{\mu_1(m\boldsymbol{I} - \boldsymbol{y}\boldsymbol{y}^\top)}{m^2} \boldsymbol{X}^\top \boldsymbol{w} \\ & - \frac{\mu_2}{m}(\boldsymbol{X}\boldsymbol{y})^\top \boldsymbol{w} + \frac{\lambda}{m}\boldsymbol{e}^\top \boldsymbol{\xi}, \\ \text{s.t. } & \boldsymbol{Y}\boldsymbol{X}^\top \boldsymbol{w} \geq \boldsymbol{e} - \boldsymbol{\xi},\ \boldsymbol{\xi} \geq \boldsymbol{0}, \end{aligned} \tag{5}$$

where $\boldsymbol{e}$ is all-one vector. By introducing the Lagrange multipliers $\boldsymbol{\alpha} \geq \boldsymbol{0}$ and $\boldsymbol{\beta} \geq \boldsymbol{0}$ for the first and the second constraints respectively, the Lagrangian of Eq. (5) leads to

$$\begin{aligned} L = & \frac{1}{2}\boldsymbol{w}^\top \boldsymbol{Q}\boldsymbol{w} - \boldsymbol{w}^\top \boldsymbol{X}\boldsymbol{Y}\left(\frac{\mu_2}{m}\boldsymbol{e} + \boldsymbol{\alpha}\right) + \boldsymbol{\alpha}^\top \boldsymbol{e} \\ & + \boldsymbol{\xi}^\top \left(\frac{\lambda}{m}\boldsymbol{e} - \boldsymbol{\alpha} - \boldsymbol{\beta}\right) \end{aligned} \tag{6}$$

where $\boldsymbol{Q} = \boldsymbol{I} + 2\mu_1 \boldsymbol{X}(m\boldsymbol{I} - \boldsymbol{y}\boldsymbol{y}^\top)\boldsymbol{X}^\top/m^2$. By setting the partial derivative of $\{\boldsymbol{w}, \boldsymbol{\xi}\}$ to zero, we have

$$\boldsymbol{w} = \boldsymbol{Q}^{-1}\boldsymbol{X}\boldsymbol{Y}\left(\frac{\mu_2}{m}\boldsymbol{e} + \boldsymbol{\alpha}\right),\ \boldsymbol{0} \leq \boldsymbol{\alpha} \leq \frac{\lambda}{m}\boldsymbol{e}. \tag{7}$$

By substituting Eq. (7) into Eq. (6), the dual of Eq. (5) can be cast as:

$$\begin{aligned} \min_{\boldsymbol{\alpha}} \ & \frac{1}{2}\left(\frac{\mu_2}{m}\boldsymbol{e} + \boldsymbol{\alpha}\right)^\top \boldsymbol{Y}\boldsymbol{X}^\top \boldsymbol{Q}^{-1}\boldsymbol{X}\boldsymbol{Y}\left(\frac{\mu_2}{m}\boldsymbol{e} + \boldsymbol{\alpha}\right) - \boldsymbol{e}^\top \boldsymbol{\alpha}, \\ \text{s.t. } & \boldsymbol{0} \leq \boldsymbol{\alpha} \leq \frac{\lambda}{m}\boldsymbol{e}. \end{aligned} \tag{8}$$

According to the Sherman-Morrison-Woodbury formula $(\boldsymbol{I} + \boldsymbol{X}\boldsymbol{A}\boldsymbol{X}^\top)^{-1} = \boldsymbol{I} - \boldsymbol{X}(\boldsymbol{A}^{-1} + \boldsymbol{X}^\top\boldsymbol{X})^{-1}\boldsymbol{X}^\top$, we have

$$\boldsymbol{Q}^{-1}\boldsymbol{X} = \boldsymbol{X}(\boldsymbol{I} + \boldsymbol{A}\boldsymbol{G})^{-1}, \tag{9}$$

where $\boldsymbol{A} = 2\mu_1(m\boldsymbol{I} - \boldsymbol{y}\boldsymbol{y}^\top)/m^2$ and $\boldsymbol{G} = \boldsymbol{X}^\top\boldsymbol{X}$ is the kernel matrix. Then $\boldsymbol{X}^\top\boldsymbol{Q}^{-1}\boldsymbol{X} = \boldsymbol{G}(\boldsymbol{I} + \boldsymbol{A}\boldsymbol{G})^{-1}$. By denoting $\boldsymbol{H} = \boldsymbol{Y}\boldsymbol{G}(\boldsymbol{I} + \boldsymbol{A}\boldsymbol{G})^{-1}\boldsymbol{Y}$, the objective function of Eq. (8) can be written as

$$f(\boldsymbol{\alpha}) = \frac{1}{2}\boldsymbol{\alpha}^\top \boldsymbol{H}\boldsymbol{\alpha} + \left(\frac{\mu_2}{m}\boldsymbol{H}\boldsymbol{e} - \boldsymbol{e}\right)^\top \boldsymbol{\alpha} + \text{const.}$$

Negelect the const term which doesn't have influence on the optimization, we arrive at the final formulation of the dual kernel ODM^L,

$$\min_{\boldsymbol{\alpha}} \ \frac{\boldsymbol{\alpha}^\top \boldsymbol{H}\boldsymbol{\alpha}}{2} + \left(\frac{\mu_2}{m}\boldsymbol{H}\boldsymbol{e} - \boldsymbol{e}\right)^\top \boldsymbol{\alpha},\ \text{s.t. } \boldsymbol{0} \leq \boldsymbol{\alpha} \leq \frac{\lambda}{m}\boldsymbol{e}. \tag{10}$$

For prediction, according to Eq. (7) and Eq. (9), one can obtain the coefficients $\boldsymbol{w}$ from the optimal $\boldsymbol{\alpha}^*$ as

$$\boldsymbol{w} = \boldsymbol{X}(\boldsymbol{I} + \boldsymbol{A}\boldsymbol{G})^{-1}\boldsymbol{Y}\left(\frac{\mu_2}{m}\boldsymbol{e} + \boldsymbol{\alpha}^*\right) = \boldsymbol{X}\boldsymbol{\nu},$$

where $\boldsymbol{\nu} = (\boldsymbol{I} + \boldsymbol{A}\boldsymbol{G})^{-1}\boldsymbol{Y}(\mu_2\boldsymbol{e}/m + \boldsymbol{\alpha}^*)$. Hence for testing instance $\boldsymbol{z}$, its label can be obtained by $\text{sign}(\boldsymbol{w}^\top\phi(\boldsymbol{z})) = \text{sign}(\sum_{i=1}^{m}\nu_i\kappa(\boldsymbol{x}_i, \boldsymbol{z}))$.

3.2 ODM

The idea of ODM^L is straightforward, however, the resultant objective function is quite complex. In this section, we try to propose a simpler but more powerful formulation.

Note that SVM set the minimum margin as 1 by scaling $\boldsymbol{w}$, in the same manner, we can also fix the margin mean as 1. Then the deviation of the margin of $(\boldsymbol{x}_i, y_i)$ to the

margin mean is $|y_i \boldsymbol{w}^\top \phi(\boldsymbol{x}_i) - 1|$. By minimizing the margin variance, we arrive at the following formulation,

$$\begin{aligned} \min_{\boldsymbol{w},\xi_i,\epsilon_i} \ & \frac{1}{2}\|\boldsymbol{w}\|^2 + \frac{\lambda}{m}\sum_{i=1}^{m}(\xi_i^2 + \epsilon_i^2), \\ \text{s.t. } & y_i \boldsymbol{w}^\top \phi(\boldsymbol{x}_i) \geq 1 - \xi_i,\ y_i \boldsymbol{w}^\top \phi(\boldsymbol{x}_i) \leq 1 + \epsilon_i,\ \forall i. \end{aligned} \tag{11}$$

where λ is trading-off parameter. The margin of $(\boldsymbol{x}_i, y_i)$ is either smaller or greater than the margin mean, so at most one of ξ_i and ϵ_i can be positive. In addition, if one is positive, the other must be zero (otherwise if it's negative, we can set it as zero without violating any constraint but decrease the objective function value), so the second term of the objective function is exactly the margin variance.

The hyperplane $y_i \boldsymbol{w}^\top \phi(\boldsymbol{x}_i) = 1$ divides the space into two subspaces. For each instance, no matter which space it lies in, it will suffer a loss which is quadratic with the deviation. However, the instances lie in the space corresponding to $y_i \boldsymbol{w}^\top \phi(\boldsymbol{x}_i) < 1$ are much easier to be misclassified than the other. So it is more reasonable to set different weights for the loss of instances in different spaces, i.e., the second term of Eq. (11) can be modified as $\frac{\lambda}{m}\sum_{i=1}^{m}(\xi_i^2 + \mu\epsilon_i^2)$, where μ is the parameter for trading-off two different deviations. According to the representer theorem [23], the optimal solution will be spanned by the support vectors. Unfortunately, for ODM, almost all training instances are support vectors. To make the solution sparse, we introduce a θ-insensitive loss like SVR, i.e., the instances whose deviation is smaller than θ are tolerated and only those whose deviation is larger than θ will suffer a loss. Finally, we obtain the formulation of ODM,

$$\begin{aligned} \min_{\boldsymbol{w},\xi_i,\epsilon_i} \ & \frac{1}{2}\|\boldsymbol{w}\|^2 + \frac{\lambda}{m}\sum_{i=1}^{m}\frac{\xi_i^2 + \mu\epsilon_i^2}{(1-\theta)^2}, \\ \text{s.t. } & y_i \boldsymbol{w}^\top \phi(\boldsymbol{x}_i) \geq 1 - \theta - \xi_i, \\ & y_i \boldsymbol{w}^\top \phi(\boldsymbol{x}_i) \leq 1 + \theta + \epsilon_i,\ \forall i. \end{aligned} \tag{12}$$

where λ and μ are described previously, θ is a parameter for controlling the number of support vectors, i.e., the sparsity of the solution, and $(1-\theta)^2$ in the denominator is to scale the second term to be a surrogate loss for 0-1 loss.

Introduce the Lagrange multipliers $\boldsymbol{\zeta} \geq \boldsymbol{0}$ and $\boldsymbol{\beta} \geq \boldsymbol{0}$ for the two constraints respectively, the Lagrangian of Eq. (12) leads to

$$\begin{aligned} L = & \frac{\|\boldsymbol{w}\|^2}{2} + \frac{\lambda\boldsymbol{\xi}^\top\boldsymbol{\xi}}{m(1-\theta)^2} + \frac{\lambda\mu\boldsymbol{\epsilon}^\top\boldsymbol{\epsilon}}{m(1-\theta)^2} - (\boldsymbol{\zeta}-\boldsymbol{\beta})^\top \boldsymbol{Y}\boldsymbol{X}^\top \boldsymbol{w} \\ & + \boldsymbol{\zeta}^\top((1-\theta)\boldsymbol{e} - \boldsymbol{\xi}) - \boldsymbol{\beta}^\top((1+\theta)\boldsymbol{e} + \boldsymbol{\epsilon}). \end{aligned} \tag{13}$$

By setting the partial derivative of $\{\boldsymbol{w}, \boldsymbol{\xi}, \boldsymbol{\epsilon}\}$ to zero, we have

$$\boldsymbol{w} = \boldsymbol{X}\boldsymbol{Y}(\boldsymbol{\zeta} - \boldsymbol{\beta}), \boldsymbol{\xi} = \frac{m(1-\theta)^2\boldsymbol{\zeta}}{2\lambda}, \boldsymbol{\epsilon} = \frac{m(1-\theta)^2\boldsymbol{\beta}}{2\lambda\mu}. \tag{14}$$

By substituting Eq. (14) into Eq. (13), we have

$$\begin{aligned} L = & -\frac{1}{2}(\boldsymbol{\zeta}-\boldsymbol{\beta})^\top \boldsymbol{Q}(\boldsymbol{\zeta}-\boldsymbol{\beta}) - \frac{m(1-\theta)^2(\mu\boldsymbol{\zeta}^\top\boldsymbol{\zeta} + \boldsymbol{\beta}^\top\boldsymbol{\beta})}{4\lambda\mu} \\ & + (1-\theta)(\boldsymbol{\zeta}^\top\boldsymbol{e} - (1+\theta)\boldsymbol{\beta}^\top\boldsymbol{e}, \end{aligned}$$

where $\boldsymbol{Q} = \boldsymbol{Y}\boldsymbol{X}^\top\boldsymbol{X}\boldsymbol{Y}$. Denote $\boldsymbol{\alpha}^\top = [\boldsymbol{\zeta}^\top, \boldsymbol{\beta}^\top]$, then $\boldsymbol{\zeta} = [\boldsymbol{I}, \boldsymbol{0}]\boldsymbol{\alpha}$, $\boldsymbol{\beta} = [\boldsymbol{0}, \boldsymbol{I}]\boldsymbol{\alpha}$ and $\boldsymbol{\zeta} - \boldsymbol{\beta} = [\boldsymbol{I}, -\boldsymbol{I}]\boldsymbol{\alpha}$. The dual of Eq. (12) can be cast as:

$$\begin{aligned} \min_{\boldsymbol{\alpha}} \ & \frac{1}{2}\boldsymbol{\alpha}^\top \begin{bmatrix} \boldsymbol{Q} + \frac{m(1-\theta)^2}{2\lambda}\boldsymbol{I} & -\boldsymbol{Q} \\ -\boldsymbol{Q} & \boldsymbol{Q} + \frac{m(1-\theta)^2}{2\lambda\mu}\boldsymbol{I} \end{bmatrix} \boldsymbol{\alpha} \\ & + \begin{bmatrix} (\theta-1)\boldsymbol{e} \\ (\theta+1)\boldsymbol{e} \end{bmatrix}^\top \boldsymbol{\alpha},\ \text{s.t. } \boldsymbol{\alpha} \geq \boldsymbol{0}. \end{aligned} \tag{15}$$

For prediction, according to Eq. (14), one can obtain the coefficients $\boldsymbol{w}$ from the optimal $\boldsymbol{\alpha}^*$ as

$$\boldsymbol{w} = \boldsymbol{X}\boldsymbol{Y}(\boldsymbol{\zeta} - \boldsymbol{\beta}) = \boldsymbol{X}\boldsymbol{Y}[\boldsymbol{I}, -\boldsymbol{I}]\boldsymbol{\alpha}^* = \boldsymbol{X}\boldsymbol{\nu},$$

where $\boldsymbol{\nu} = \boldsymbol{Y}[\boldsymbol{I}, -\boldsymbol{I}]\boldsymbol{\alpha}^*$. Hence for testing instance $\boldsymbol{z}$, its label is $\operatorname{sign}(\boldsymbol{w}^\top\phi(\boldsymbol{z})) = \operatorname{sign}(\sum_{i=1}^{m}\nu_i\kappa(\boldsymbol{x}_i, \boldsymbol{z}))$.

3.3 Optimization

Note that Eq. (10) and Eq. (15) are both the special cases of the following form, which has convex quadratic objective function and simple decoupled box constraints,

$$\min_{\boldsymbol{\alpha}} \ f(\boldsymbol{\alpha}) = \frac{1}{2}\boldsymbol{\alpha}^\top\boldsymbol{H}\boldsymbol{\alpha} + \boldsymbol{q}^\top\boldsymbol{\alpha},\ \text{s.t. } \boldsymbol{0} \leq \boldsymbol{\alpha} \leq \boldsymbol{u},$$

where $\boldsymbol{u} = \boldsymbol{\infty}$ for ODM. As suggested by [24], it can be efficiently solved by the dual coordinate descent method. In dual coordinate descent method [25], one of the variables is selected to minimize while the other variables are kept as constants at each iteration, and a closed-form solution can be achieved at each iteration. Specifically, to minimize α_i by keeping the other $\alpha_{j\neq i}$ as constants, one needs to solve the following subproblem,

$$\min_{t} \ f(\boldsymbol{\alpha} + t\boldsymbol{e}_i) \quad \text{s.t. } 0 \leq \alpha_i + t \leq u_i, \tag{16}$$

where $\boldsymbol{e}_i$ denotes the vector with 1 in the i-th coordinate and 0 elsewhere. Let $\boldsymbol{H} = [h_{ij}]_{i,j=1,\ldots,m}$, we have

$$f(\boldsymbol{\alpha} + t\boldsymbol{e}_i) = \frac{1}{2}h_{ii}t^2 + [\nabla f(\boldsymbol{\alpha})]_i t + f(\boldsymbol{\alpha}),$$

where $[\nabla f(\boldsymbol{\alpha})]_i$ is the i-th component of the gradient $\nabla f(\boldsymbol{\alpha})$. Considering that $f(\boldsymbol{\alpha} + t\boldsymbol{e}_i)$ is a simple quadratic function of t, and further note the box constraint $0 \leq \alpha_i \leq u_i$, the minimizer of Eq. (16) leads to a closed-form solution: $\alpha_i^{new} = \min(\max(\alpha_i - [\nabla f(\boldsymbol{\alpha})]_i/h_{ii}, 0), u_i)$. Algorithm 1 summarizes the pseudo-code of ODM$_{dcd}$ for kernel ODM^L and ODM.

Algorithm 1 ODM$_{dcd}$

Input: data set $\boldsymbol{X}$.
Initialize $\boldsymbol{\alpha} = \boldsymbol{0}$, calculate $\boldsymbol{H}$ and $\boldsymbol{q}$.
while $\boldsymbol{\alpha}$ not converge **do**
  for $i = 1, \ldots m$ **do**
    $[\nabla f(\boldsymbol{\alpha})]_i \leftarrow [\boldsymbol{H}\boldsymbol{\alpha} + \boldsymbol{q}]_i$.
    $\alpha_i \leftarrow \min\left(\max\left(\alpha_i - \frac{[\nabla f(\boldsymbol{\alpha})]_i}{h_{ii}}, 0\right), u_i\right)$.
  end for
end while
Output: $\boldsymbol{\alpha}$.

3.3.1 Speedup for Linear Kernel

ODM$_{dcd}$ can efficiently deal with kernel ODM^L and ODM. However, the inherent computational cost for the kernel matrix takes $O(m^2)$ time, which might be computational prohibitive for large scale problems. To make them more useful, in the following, we present a fast linear kernel ODM^L and ODM for large scale problems.

For linear kernel ODM^L and ODM, the objective function of primal problem Eq. (4) and Eq. (12) can be reformulated as the following form, respectively,

$$f_L(\boldsymbol{w}) = \frac{\|\boldsymbol{w}\|^2}{2} + \frac{\mu_1 \boldsymbol{w}^\top \boldsymbol{X}\boldsymbol{X}^\top \boldsymbol{w}}{m} - \frac{\mu_1 \boldsymbol{w}^\top \boldsymbol{X}\boldsymbol{y}\boldsymbol{y}^\top \boldsymbol{X}^\top \boldsymbol{w}}{m^2} - \frac{\mu_2}{m}(\boldsymbol{X}\boldsymbol{y})^\top \boldsymbol{w} + \frac{\lambda}{m}\sum_{i=1}^{m} \max\{0, 1 - y_i \boldsymbol{w}^\top \boldsymbol{x}_i\}, \tag{17}$$

$$f_O(\boldsymbol{w}) = \frac{\|\boldsymbol{w}\|^2}{2} + \frac{\lambda \sum_{i=1}^{m} \max\{0, 1 - \theta - y_i \boldsymbol{w}^\top \boldsymbol{x}_i\}^2}{m(1-\theta)^2} + \frac{\lambda\mu \sum_{i=1}^{m} \max\{0, y_i \boldsymbol{w}^\top \boldsymbol{x}_i - 1 - \theta\}^2}{m(1-\theta)^2}. \tag{18}$$

For large scale problems, the computational cost for the gradient of (17) and (18) is expensive because its computation involves all the training instances. Stochastic gradient descent (SGD) works by computing a noisy unbiased estivion of the gradient via sampling a subset of the training instances [26]. It can be theoretically proven that when the objective is convex, SGD converges to the global optimal solution in expectation [27], [28].

The following theorem presents an approach to obtain an unbiased estimation of the gradient $\nabla f_L(\boldsymbol{w})$ and $\nabla f_O(\boldsymbol{w})$.

Theorem 3.1. *If two instances $(\boldsymbol{x}_i, y_i)$ and $(\boldsymbol{x}_j, y_j)$ are randomly sampled from the training set independently, then*

$$\nabla f_L(\boldsymbol{w}, \boldsymbol{x}_i, \boldsymbol{x}_j) = \boldsymbol{w} + 2\mu_1 \boldsymbol{x}_i \boldsymbol{x}_i^\top \boldsymbol{w} - 2\mu_1 y_i y_j \boldsymbol{x}_i \boldsymbol{x}_j^\top \boldsymbol{w} - \mu_2 y_i \boldsymbol{x}_i - \lambda y_i \boldsymbol{x}_i 1_{i\in I_1},$$

$$\nabla f_O(\boldsymbol{w}, \boldsymbol{x}_i) = \boldsymbol{w} + \frac{2\lambda(y_i \boldsymbol{w}^\top \boldsymbol{x}_i + \theta - 1) y_i \boldsymbol{x}_i 1_{i\in I_2}}{(1-\theta)^2} + \frac{2\lambda\mu(y_i \boldsymbol{w}^\top \boldsymbol{x}_i - \theta - 1) y_i \boldsymbol{x}_i 1_{i\in I_3}}{(1-\theta)^2},$$

are the unbiased estimation of $\nabla f_L(\boldsymbol{w})$ and $\nabla f_O(\boldsymbol{w})$ respectively, where $I_1 = \{i \mid y_i \boldsymbol{w}^\top \boldsymbol{x}_i < 1\}$, $I_2 = \{i \mid y_i \boldsymbol{w}^\top \boldsymbol{x}_i < 1-\theta\}$, $I_3 = \{i \mid y_i \boldsymbol{w}^\top \boldsymbol{x}_i > 1+\theta\}$ are index sets.

Proof. The gradient of $f_L(\boldsymbol{w})$ is

$$\nabla f_L(\boldsymbol{w}) = \boldsymbol{w} + \frac{2\mu_1}{m}\boldsymbol{X}\boldsymbol{X}^\top \boldsymbol{w} - \frac{2\mu_1}{m^2}\boldsymbol{X}\boldsymbol{y}\boldsymbol{y}^\top \boldsymbol{X}^\top \boldsymbol{w} - \frac{\mu_2}{m}\boldsymbol{X}\boldsymbol{y} - \frac{\lambda}{m}\sum_{i=1}^{m} y_i \boldsymbol{x}_i 1_{i\in I_1}.$$

Note that $\mathbb{E}[\boldsymbol{x}_i \boldsymbol{x}_i^\top] = \boldsymbol{X}\boldsymbol{X}^\top/m$ and $\mathbb{E}[y_i \boldsymbol{x}_i] = \boldsymbol{X}\boldsymbol{y}/m$, with the linearity of expectation and the independence between $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$, we have $\mathbb{E}[\nabla f_L(\boldsymbol{w}, \boldsymbol{x}_i, \boldsymbol{x}_j)] = \nabla f_L(\boldsymbol{w})$, thus $\nabla f_L(\boldsymbol{w}, \boldsymbol{x}_i, \boldsymbol{x}_j)$ is an unbiased gradient of $f_L(\boldsymbol{w})$.

Again the gradient of $f_O(\boldsymbol{w})$ is

$$\nabla f_O(\boldsymbol{w}) = \boldsymbol{w} + \frac{2\lambda \sum_{i=1}^{m}(y_i \boldsymbol{w}^\top \boldsymbol{x}_i + \theta - 1) y_i \boldsymbol{x}_i 1_{i\in I_2}}{m(1-\theta)^2} + \frac{2\lambda \sum_{i=1}^{m} \mu(y_i \boldsymbol{w}^\top \boldsymbol{x}_i - \theta - 1) y_i \boldsymbol{x}_i 1_{i\in I_3}}{m(1-\theta)^2}.$$

According to the linearity of expectation, we have $\mathbb{E}[\nabla f_O(\boldsymbol{w}, \boldsymbol{x}_i)] = \nabla f_O(\boldsymbol{w})$, thus $\nabla f_O(\boldsymbol{w}, \boldsymbol{x}_i)$ is an unbiased gradient of $f_O(\boldsymbol{w})$. □

Based on Theorem 3.1, the stochastic gradient update can be formed as

$$\boldsymbol{w}_{t+1} = \boldsymbol{w}_t - \eta_t \boldsymbol{g}_t, \tag{19}$$

where $\boldsymbol{g}_t$ is $\nabla f_L(\boldsymbol{w}, \boldsymbol{x}_i, \boldsymbol{x}_j)$ for ODM^L and $\nabla f_O(\boldsymbol{w}, \boldsymbol{x}_i)$ for ODM respectively, and η_t is a suitably chosen step size parameter in the t-th iteration.

Since the objective function of ODM is differentiable, in practice we use the SVRG method which is more stable than SGD [29]. Besides performing the normal stochastic gradient update (19) at each iteration, it also occasionally compute full gradient, which can be used to reduce the variance of the stochastic gradient estimation. Algorithm 2 summarizes the pseudo-code of ODM$_{svrg}$.

Algorithm 2 ODM$_{svrg}$

Input: data set $\boldsymbol{X}$.
Initialize $\bar{\boldsymbol{w}}_0 = \boldsymbol{0}$.
for $s = 1, 2, \ldots$ **do**
  $\boldsymbol{w}_0 = \bar{\boldsymbol{w}} = \bar{\boldsymbol{w}}_{s-1}$.
  Compute full gradient $\bar{\boldsymbol{\mu}}$.
  for $t = 1, 2, \ldots, m$ **do**
    Randomly sample training instance $(\boldsymbol{x}_i, y_i)$.
    Compute the unbiased stochastic gradient $\boldsymbol{g}_t$.
    $\boldsymbol{w}_t = \boldsymbol{w}_{t-1} - \eta(\nabla f_O(\boldsymbol{w}_{t-1}, \boldsymbol{x}_i) - \nabla f_O(\bar{\boldsymbol{w}}, \boldsymbol{x}_i) + \bar{\boldsymbol{\mu}})$.
  end for
  Set $\bar{\boldsymbol{w}}_s = \boldsymbol{w}_t$ for randomly chosen $t \in [m]$.
end for
Output: $\bar{\boldsymbol{w}}$

4 MCODM

In this section, we try to extend the binary ODM described in Section 3.2 to multi-class version [30]. By replacing the margin term $y_i \boldsymbol{w}^\top \phi(\boldsymbol{x}_i)$ in Eq. (12) with the general margin defined in Eq. (1), we can achieve the multi-class version of kernel ODM, i.e., kernel mcODM:

$$\begin{aligned} \min_{\boldsymbol{w}_l, \xi_i, \epsilon_i} \ & \frac{1}{2}\sum_{l=1}^{k} \|\boldsymbol{w}_l\|^2 + \frac{\lambda}{m}\sum_{i=1}^{m} \frac{\xi_i^2 + \mu\epsilon_i^2}{(1-\theta)^2}, \\ \text{s.t. } & \boldsymbol{w}_{y_i}^\top \phi(\boldsymbol{x}_i) - \max_{l \neq y_i} \boldsymbol{w}_l^\top \phi(\boldsymbol{x}_i) \geq 1 - \theta - \xi_i, \\ & \boldsymbol{w}_{y_i}^\top \phi(\boldsymbol{x}_i) - \max_{l \neq y_i} \boldsymbol{w}_l^\top \phi(\boldsymbol{x}_i) \leq 1 + \theta + \epsilon_i, \ \forall i, \end{aligned}$$

where the regularization term turns into summation of norm of all the scoring functions.

Due to the max operator in the second constraint, mcODM is a non-differentiable non-convex programming, which is quite difficult to solve directly. We first relax mcODM into a series of convex QP, and then extend the block coordinate descent method [31] to solve the dual of each QP.

At each iteration, we recast the first constraint as $k-1$ linear inequality constraints $\boldsymbol{w}_{y_i}^\top \phi(\boldsymbol{x}_i) - \boldsymbol{w}_l^\top \phi(\boldsymbol{x}_i) \geq 1 - \theta - \xi_i$, $l \neq y_i$ and replace the second constraint with $\boldsymbol{w}_{y_i}^\top \phi(\boldsymbol{x}_i) - M_i \leq 1 + \theta + \epsilon_i$, where $M_i = \max_{l \neq y_i} \bar{\boldsymbol{w}}_l^\top \phi(\boldsymbol{x}_i)$ and $\bar{\boldsymbol{w}}_l$ is the

solution to the previous iteration. Then we can repeatedly solve the following convex QP problem until convergence:

$$\begin{aligned}\min_{\boldsymbol{w}_l,\xi_i,\epsilon_i}\ & \frac{1}{2}\sum_{l=1}^{k}\|\boldsymbol{w}_l\|^2+\frac{\lambda}{m}\sum_{i=1}^{m}\frac{\xi_i^2+\mu\epsilon_i^2}{(1-\theta)^2},\\ \text{s.t. }& \boldsymbol{w}_{y_i}^\top\phi(\boldsymbol{x}_i)-\boldsymbol{w}_l^\top\phi(\boldsymbol{x}_i)\geq 1-\theta-\xi_i,\ \forall l\neq y_i,\\ & \boldsymbol{w}_{y_i}^\top\phi(\boldsymbol{x}_i)-M_i\leq 1+\theta+\epsilon_i,\ \forall i.\end{aligned}\tag{20}$$

Introduce the Lagrange multipliers $\zeta_i^l\geq 0, l\neq y_i$ for the first $k-1$ constraints and $\beta_i\geq 0$ for the last constraint respectively, the Lagrangian function of Eq. (20) leads to

$$\begin{aligned}L=&\frac{1}{2}\sum_{l=1}^{k}\|\boldsymbol{w}_l\|^2+\frac{\lambda}{m}\sum_{i=1}^{m}\frac{\xi_i^2+\mu\epsilon_i^2}{(1-\theta)^2}\\ &-\sum_{i=1}^{m}\sum_{l\neq y_i}\zeta_i^l(\boldsymbol{w}_{y_i}^\top\phi(\boldsymbol{x}_i)-\boldsymbol{w}_l^\top\phi(\boldsymbol{x}_i)-1+\theta+\xi_i)\\ &+\sum_{i=1}^{m}\beta_i(\boldsymbol{w}_{y_i}^\top\phi(\boldsymbol{x}_i)-M_i-1-\theta-\epsilon_i),\end{aligned}$$

By setting the partial derivations of variables $\{\boldsymbol{w}_l,\xi_i,\epsilon_i\}$ to zero, we have

$$\begin{aligned}\boldsymbol{w}_l&=\sum_{i=1}^{m}\left(\delta_{y_i,l}\sum_{s\neq y_i}\zeta_i^s-(1-\delta_{y_i,l})\zeta_i^l-\delta_{y_i,l}\beta_i\right)\phi(\boldsymbol{x}_i),\\ \xi_i&=\frac{m(1-\theta)^2}{2\lambda}\sum_{l\neq y_i}\zeta_i^l,\ \epsilon_i=\frac{m(1-\theta)^2}{2\lambda\mu}\beta_i.\end{aligned}\tag{21}$$

where $\delta_{y_i,l}$ equals 1 when $y_i=l$ and 0 otherwise. By further defining $\alpha_i^l=-\zeta_i^l$ for $\forall l\neq y_i$ and $\alpha_i^{y_i}=\sum_{s\neq y_i}\zeta_i^s$, we have

$$\boldsymbol{w}_l=\sum_{i=1}^{m}(\alpha_i^l-\delta_{y_i,l}\beta_i)\phi(\boldsymbol{x}_i).\tag{22}$$

By substituting Eq. (22) and Eq. (21) into the Lagrangian function, we can obtain the following dual problem

$$\begin{aligned}\min_{\alpha_i^l,\alpha_i^{y_i},\beta_i}\ & \frac{1}{2}\sum_{l=1}^{k}\|\boldsymbol{w}_l\|^2+\frac{m(1-\theta)^2}{4\lambda\mu}\left(\mu\sum_{i=1}^{m}(\alpha_i^{y_i})^2+\sum_{i=1}^{m}\beta_i^2\right)\\ &+(1-\theta)\sum_{i=1}^{m}\sum_{l\neq y_i}\alpha_i^l+(M_i+1+\theta)\sum_{i=1}^{m}\beta_i,\\ \text{s.t. }&\sum_{l=1}^{k}\alpha_i^l=0,\ \alpha_i^l\leq 0,\ \beta_i\geq 0,\ \forall l\neq y_i,\ \forall i.\end{aligned}\tag{23}$$

The objective function in Eq. (23) involves $m(k+1)$ variables in total, so it is not easy to optimize with respect to all the variables simultaneously. Note that all the constraints can be partitioned into m disjoint sets, and the i-th set only involves $\alpha_i^1,\ldots,\alpha_i^k,\beta_i$, so the variables can be divided into m decoupled groups and an efficient block coordinate descent algorithm can be applied. Specifically, we sequentially select a group of $k+1$ variables $\alpha_i^1,\ldots,\alpha_i^k,\beta_i$ associated with instance $\boldsymbol{x}_i$ to minimize, while keeping other variables as constants, and repeat this procedure until convergence. Algorithm 3 below details the kernel mcODM.

Algorithm 3 Kernel mcODM

1: **Input:** data set $\boldsymbol{X}$.
2: Initialize $\boldsymbol{\alpha}^\top=[\alpha_1^1,\ldots,\alpha_1^k,\ldots,\alpha_m^1,\ldots,\alpha_m^k]$ and $\boldsymbol{\beta}^\top=[\beta_1,\ldots,\beta_m]$ as zero vector.
3: **while** $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ not converge **do**
4: **for** $i=1,\ldots,m$ **do**
5: $M_i\leftarrow\max_{l\neq y_i}\sum_{j=1}^{m}(\alpha_j^l-\delta_{y_j,l}\beta_j)\kappa(\boldsymbol{x}_j,\boldsymbol{x}_i)$.
6: **end for**
7: Solve Eq. (23) by block coordinate descent method.
8: **end while**
9: **Output:** $\boldsymbol{\alpha}$, $\boldsymbol{\beta}$.

4.1 Sub-problem

The sub-problem in step 7 of Algorithm 3 is also a convex QP with $k+1$ variables, which can be accomplished by some standard QP solvers. However, by exploiting its special structure, i.e., only a small quantity of cross terms are involved, we can derive an algorithm to solve this sub-problem just by sorting, which can be much faster than general QP solvers.

Note that all variables except $\alpha_i^1,\ldots,\alpha_i^k,\beta_i$ are fixed, so we have the following sub-problem:

$$\begin{aligned}\min_{\alpha_i^l,\alpha_i^{y_i},\beta_i}\ & \sum_{l\neq y_i}\frac{A}{2}(\alpha_i^l)^2+\sum_{l\neq y_i}B_l\alpha_i^l+\frac{D}{2}(\alpha_i^{y_i})^2-A\alpha_i^{y_i}\beta_i\\ &+B_{y_i}\alpha_i^{y_i}+\frac{E}{2}\beta_i^2+F\beta_i,\\ \text{s.t. }&\sum_{l=1}^{k}\alpha_i^l=0,\ \alpha_i^l\leq 0,\ \beta_i\geq 0,\ \forall l\neq y_i.\end{aligned}\tag{24}$$

where $A=\kappa(\boldsymbol{x}_i,\boldsymbol{x}_i)$, $B_l=\sum_{j\neq i}\kappa(\boldsymbol{x}_i,\boldsymbol{x}_j)(\alpha_j^l-\delta_{y_j,l}\beta_j)+1-\theta$ for $\forall l\neq y_i$, $B_{y_i}=\sum_{j\neq i}\kappa(\boldsymbol{x}_i,\boldsymbol{x}_j)(\alpha_j^{y_i}-\delta_{y_j,y_i}\beta_j)$, $D=A+m(1-\theta)^2/2\lambda$, $E=A+m(1-\theta)^2/2\lambda\mu$ and $F=M_i+1+\theta-B_{y_i}$.

The KKT conditions of Eq. (24) indicate that there are scalars ν, ρ_l and η such that

$$\sum_{l=1}^{k}\alpha_i^l=0,\tag{25}$$
$$\alpha_i^l\leq 0,\ \forall l\neq y_i,\tag{26}$$
$$\beta_i\geq 0,\tag{27}$$
$$\rho_l\alpha_i^l=0,\ \rho_l\geq 0,\ \forall l\neq y_i,\tag{28}$$
$$A\alpha_i^l+B_l-\nu+\rho_l=0,\ \forall l\neq y_i,\tag{29}$$
$$\eta\beta_i=0,\ \eta\geq 0,\tag{30}$$
$$-A\alpha_i^{y_i}+E\beta_i+F-\eta=0,\tag{31}$$
$$D\alpha_i^{y_i}-A\beta_i+B_{y_i}-\nu=0.\tag{32}$$

According to Eq. (26), Eq. (28) and Eq. (29) are equivalent to

$$A\alpha_i^l+B_l-\nu=0,\text{ if }\alpha_i^l<0,\ \forall l\neq y_i,\tag{33}$$
$$B_l-\nu\leq 0,\text{ if }\alpha_i^l=0,\ \forall l\neq y_i.\tag{34}$$

In the same manner, Eq. (30) and Eq. (31) are equivalent to

$$-A\alpha_i^{y_i}+E\beta_i+F=0,\text{ if }\beta_i>0,\tag{35}$$
$$-A\alpha_i^{y_i}+F\geq 0,\text{ if }\beta_i=0.\tag{36}$$

Thus KKT conditions turn to Eq. (25) - Eq. (27) and Eq. (32) - Eq. (36). Note that

$$\alpha_i^l = \min\left(0, \frac{\nu - B_l}{A}\right), \ \forall l \neq y_i, \tag{37}$$

satisfies KKT conditions Eq. (26) and Eq. (33) - Eq. (34) and

$$\beta_i = \max\left(0, \frac{A\alpha_i^{y_i} - F}{E}\right), \tag{38}$$

satisfies KKT conditions Eq. (27) and Eq. (35) - Eq. (36). By substituting Eq. (38) into Eq. (32), we obtain

$$D\alpha_i^{y_i} + B_{y_i} - \nu = \max\left(0, \frac{A}{E}(A\alpha_i^{y_i} - F)\right). \tag{39}$$

Now we need consider the following two cases:

1) $A\alpha_i^{y_i} \leq F$, according to Eq. (38) and Eq. (39), we have $\beta_i = 0$ and $\alpha_i^{y_i} = (\nu - B_{y_i})/D$. Thus, $A(\nu - B_{y_i})/D \leq F$, which implies that $\nu \leq B_{y_i} + DF/A$.
2) $A\alpha_i^{y_i} > F$, according to Eq. (38) and Eq. (39), we have $\beta_i = (A\alpha_i^{y_i} - F)/E > 0$ and $\alpha_i^{y_i} = (E\nu - AF - EB_{y_i})/(DE - A^2)$. Thus, $A(E\nu - AF - EB_{y_i})/(DE - A^2) > F$, which implies that $\nu > B_{y_i} + DF/A$.

The remaining task is to find ν such that Eq. (25) holds. With Eq. (25) and Eq. (37), it can be shown that

$$\nu = \frac{\frac{AB_{y_i}}{D} + \sum_{l:\alpha_i^l<0} B_l}{\frac{A}{D} + |\{l \mid \alpha_i^l < 0\}|}, \text{ if } A\alpha_i^{y_i} \leq F, \tag{40}$$

$$\nu = \frac{\frac{AEB_{y_i} + A^2F}{DE - A^2} + \sum_{l:\alpha_i^l<0} B_l}{\frac{AE}{DE - A^2} + |\{l \mid \alpha_i^l < 0\}|}, \text{ if } A\alpha_i^{y_i} > F. \tag{41}$$

In both cases, the optimal ν takes the form of $(P + \sum_{l:\alpha_i^l<0} B_l)/(Q + |\{l \mid \alpha_i^l < 0\}|)$, where P and Q are some constants. [32] showed that it can be found by sorting $\{B_l\}$ for $\forall l \neq y_i$ in decreasing order and then sequentially adding them into an empty set Φ, until

$$\nu^* = \frac{P + \sum_{l\in\Phi} B_l}{Q + |\Phi|} \geq \max_{l\notin\Phi} B_l.$$

Note that the Hessian matrix of the objective function of Eq. (24) is positive definite, which guarantees the existence and uniqueness of the optimal solution, so only one of Eq. (40) and Eq. (41) can hold. We can first compute ν^* according to Eq. (40), and then check whether the constraint of ν is satisfied. If not, we further compute ν^* according to Eq. (41). Algorithm 4 summarizes the pseudo-code for solving the sub-problem.

For prediction, according to Eq. (22), we have

$$\boldsymbol{w}_l = \sum_{i=1}^{m}(\alpha_i^l - \delta_{y_i,l}\beta_i)\phi(\boldsymbol{x}_i) = \boldsymbol{X}\boldsymbol{\nu}_l,$$

where $\boldsymbol{\nu}_l^\top = [\alpha_1^l - \delta_{y_1,l}\beta_1, \ldots, \alpha_m^l - \delta_{y_m,l}\beta_m]$. Hence for testing instance $\boldsymbol{z}$, its label can be obtained by

$$\operatorname*{argmax}_{l\in[k]}\left(\boldsymbol{w}_l^\top\phi(\boldsymbol{z})\right) = \operatorname*{argmax}_{l\in[k]}\left(\sum_{i=1}^{m}[\boldsymbol{\nu}_l]_i\kappa(\boldsymbol{x}_i, \boldsymbol{z})\right),$$

where $[\boldsymbol{\nu}_l]_i$ is the i-th component of $\boldsymbol{\nu}_l$.

Algorithm 4 Solving the sub-problem

1: **Input:** Parameters A, $B = \{B_1, \ldots, B_k\}$, D, E, F.
2: Initialize $\hat{B} \leftarrow B$, then swap $\hat{B}_1$ and $\hat{B}_{y_i}$, and sort $\hat{B}\backslash\{\hat{B}_1\}$ in decreasing order.
3: $i \leftarrow 2$, $\nu \leftarrow AB_{y_i}/D$.
4: **while** $i \leq k$ and $\nu/(i - 2 + A/D) < \hat{B}_i$ **do**
5: $\quad\nu \leftarrow \nu + \hat{B}_i$.
6: $\quad i \leftarrow i + 1$.
7: **end while**
8: **if** $\nu \leq B_{y_i} + DF/A$ **then**
9: $\quad\alpha_i^l \leftarrow \min(0, (\nu - B_l)/A), \ l \neq y_i$.
10: $\quad\alpha_i^{y_i} \leftarrow (\nu - B_{y_i})/D$.
11: $\quad\beta_i \leftarrow 0$.
12: **else**
13: $\quad i \leftarrow 2$, $\nu \leftarrow (AE\hat{B}_1 + A^2F)/(DE - A^2)$.
14: $\quad$**while** $i \leq k$ and $\nu/(i - 2 + AE/(DE - A^2)) < \hat{B}_i$ **do**
15: $\quad\quad\nu \leftarrow \nu + \hat{B}_i$.
16: $\quad\quad i \leftarrow i + 1$.
17: $\quad$**end while**
18: $\quad\alpha_i^l \leftarrow \min(0, (\nu - B_l)/A), \ l \neq y_i$.
19: $\quad\alpha_i^{y_i} \leftarrow (E\nu - AF - EB_{y_i})/(DE - A^2)$.
20: $\quad\beta_i \leftarrow (A\alpha_i^{y_i} - F)/E$.
21: **end if**
22: **Output:** $\alpha_i^1, \ldots, \alpha_i^k, \beta_i$.

4.2 Speedup for Linear Kernel

In section 4.1, the proposed method can efficiently deal with kernel mcODM. However, the computation of M_i in step 5 of Algorithm 3 and the computation of parameters B_l in Algorithm 4 both involve the kernel matrix, whose inherent computational cost takes $O(m^2)$ time, so it might be computational prohibitive for large scale problems.

When linear kernel is used, these problems can be alleviated. According to Eq. (22), $\boldsymbol{w}_1, \ldots, \boldsymbol{w}_k$ is spanned by the training instance so it lies in a finite dimensional space under this circumstance. By storing $\boldsymbol{w}_1, \ldots, \boldsymbol{w}_k$ explicitly, the computational cost of $M_i = \max_{l\neq y_i} \boldsymbol{w}_l^\top \boldsymbol{x}_i$ can be much less. Moreover, note that $B_l = \sum_{j\neq i} \boldsymbol{x}_i^\top \boldsymbol{x}_j(\alpha_j^l - \delta_{y_j,l}\beta_j) = \sum_{j=1}^{m} \boldsymbol{x}_i^\top \boldsymbol{x}_j(\bar{\alpha}_j^l - \delta_{y_j,l}\bar{\beta}_j) - \boldsymbol{x}_i^\top \boldsymbol{x}_i(\bar{\alpha}_i^l - \delta_{y_i,l}\bar{\beta}_i) = \boldsymbol{w}_l^\top \boldsymbol{x}_i - A(\alpha_i^l - \delta_{y_i,l}\beta_i)$, so B_l can also be computed efficiently.

5 Theoretical Analysis

In this section, we study the statistical property of ODM. To present the generalization bound of ODM, we need to introduce the following loss function Φ,

$$\Phi(z) = 1_{z\leq 0} + \frac{(z - 1 + \theta)^2}{(1 - \theta)^2}1_{0<z\leq 1-\theta},$$

$$\gamma_{h,\theta}(\boldsymbol{x}, y) = \boldsymbol{w}_y^\top\phi(\boldsymbol{x}) - \max_{l\in\mathcal{Y}}\{\boldsymbol{w}_l^\top\phi(\boldsymbol{x}) - (1 - \theta)1_{l=y}\},$$

As can be seen, $\gamma_{h,\theta}(\boldsymbol{x}, y)$ is a lower bound of $\gamma_h(\boldsymbol{x}, y)$ and $\Phi(\gamma_{h,\theta}(\boldsymbol{x}, y)) = \Phi(\gamma_h(\boldsymbol{x}, y))$.

Theorem 5.1. *Let $\mathcal{H} = \{(\boldsymbol{x}, y) \mapsto \boldsymbol{w}_y^\top\phi(\boldsymbol{x}) \mid \sum_{l=1}^{k} \|\boldsymbol{w}_l\|^2 \leq \Lambda^2\}$ be the hypotheses space of ODM, where $\phi : \mathcal{X} \mapsto \mathbb{H}$ is a feature mapping induced by some positive definite kernel κ. Assume that $\mathcal{S} \subseteq \{\boldsymbol{x} \mid \kappa(\boldsymbol{x}, \boldsymbol{x}) \leq r^2\}$, then for any $\delta > 0$,*

with probability at least $1-\delta$, the following generalization bound holds for any $h \in \mathcal{H}$,

$$R(h) \leq \frac{1}{m}\sum_{i=1}^{m}\Phi(\gamma_h(\boldsymbol{x}_i, y_i)) + \frac{16r\Lambda}{1-\theta}\sqrt{\frac{2\pi k}{m}} + 3\sqrt{\frac{\ln(2/\delta)}{2m}}.$$

Proof. Since $\gamma_{h,\theta}(\boldsymbol{x}, y)$ is a lower bound of $\gamma_h(\boldsymbol{x}, y)$, thus

$$R(h) = \mathbb{E}[1_{\gamma_h(\boldsymbol{x},y)\leq 0}] \leq \mathbb{E}[1_{\gamma_{h,\theta}(\boldsymbol{x},y)\leq 0}] \leq \mathbb{E}[\Phi(\gamma_{h,\theta}(\boldsymbol{x}, y))].$$

Next we bound the $\mathbb{E}[\Phi(\gamma_{h,\theta}(\boldsymbol{x}, y))]$. Let $\tilde{\mathcal{H}}_\theta = \{(\boldsymbol{x}, y) \mapsto \gamma_{h,\theta}(\boldsymbol{x}, y) \mid h \in \mathcal{H}\}$, with McDiarmid inequality [33], we have the following inequality with probability at least $1-\delta$,

$$\mathbb{E}[\Phi(\gamma_{h,\theta}(\boldsymbol{x}, y))] \leq \frac{1}{m}\sum_{i=1}^{m}\Phi(\gamma_{h,\theta}(\boldsymbol{x}_i, y_i)) + 2\mathcal{R}_{\mathcal{S}}(\Phi \circ \tilde{\mathcal{H}}_\theta) + 3\sqrt{\frac{\ln(2/\delta)}{2m}},\ \forall h \in \tilde{\mathcal{H}}_\theta,$$

where $\mathcal{R}_{\mathcal{S}}$ is the Rademacher complexity. Note that $\Phi(z)$ is $\frac{2}{1-\theta}$-Lipschitz function, which yields that $\mathcal{R}_{\mathcal{S}}(\Phi \circ \tilde{\mathcal{H}}_\theta) \leq \frac{2}{1-\theta}\mathcal{R}_{\mathcal{S}}(\tilde{\mathcal{H}}_\theta)$ according to the Talagrand's lemma [34]. Further note that $\Phi(\gamma_{h,\theta}(\boldsymbol{x}, y)) = \Phi(\gamma_h(\boldsymbol{x}, y))$, thus we have

$$R(h) \leq \frac{1}{m}\sum_{i=1}^{m}\Phi(\gamma_h(\boldsymbol{x}_i, y_i)) + \frac{4\mathcal{R}_{\mathcal{S}}(\tilde{\mathcal{H}}_\theta)}{1-\theta} + 3\sqrt{\frac{\ln(2/\delta)}{2m}}.$$

The remain is to bound $\mathcal{R}_{\mathcal{S}}(\tilde{\mathcal{H}}_\theta)$. To get a tight dependence on the class size k, we apply the Theorem 7 of [35], which proves $\mathcal{R}_{\mathcal{S}}(\tilde{\mathcal{H}}_\theta) \leq 4r\Lambda\sqrt{2\pi k/m}$ by exploiting Gaussian complexity, therefore the stated result holds. □

Theorem 5.1 shows that the generalization of ODM can be upper bounded by the sum of three components, among which, the first term is the average of empirical loss of ODM, and the last term is a by-product of the McDiarmid inequality which can be ignored, so the main inspiration comes from the second term. It's not difficult to find that the smaller $r\Lambda$ and the larger $1-\theta$, the smaller this term, so that the tighter the bound. Note that any margin $\gamma \leq 2\|\boldsymbol{w}\|\|\phi(\boldsymbol{x})\| \leq 2r\Lambda$, thus to achieve good generalization performance, we should minimize the upper bound of margin. Besides, $1-\theta$ is the lower bound of the zero loss band of ODM, which is related to the minimum margin. Hence maximizing the lower bound of the margin can result in good generalization performance. Put both together, then it is verified that minimizing the margin variance when the margin mean is fixed can yield better generalization performance, which is also consistent with the previous work [11].

6 Related Works

There are a few studies considered the effect of the margin distribution on the generalization ability of SVM-style algorithms. [36] proposed the Margin Distribution Optimization (MDO) algorithm, which optimizes margin distribution by minimizing the sum of exponential loss. Specifically, the loss associated to each instance is a exponential function of the margin, and the larger the margin, the smaller the loss, and vice verse. MDO can be viewed as a method of optimizing the combination of weighted margin. However, this method tends to get a local minima with slow convergence since the objective function is non-differential and non-convex.

[37] proposed the Maximal Average Margin for Classifiers (MAMC), which directly optimizes the first moment of the margin distribution. In this case, the problem can be solved efficiently since computing the model has linear time complexity, however, this method tends to get trivial solution when the classes are not with equal size. In addition, it can be viewed as a special case of our ODM^L by ignoring margin variance.

[38] proposed a Kernel Method for the direct Optimization of the Margin Distribution (KM-OMD) from a game theoretical perspective. This method also optimizes a weighted combination of margins, where the coefficients are determined implicitly via some constraints related to the entropy of the distribution. Actually, this is equivalent to adding a scaled identity matrix to the kernel matrix of the dual of hard-margin SVM, so it can be viewed as to design a specific kernel mapping [39] and only consider a single-point margin. It is noteworthy that it contains equality constraints so only SMO-like solver can be applied, which makes it difficult for large-scale data sets.

7 Experiments

In this section, we empirically evaluate the effectiveness of our methods on both binary classification problems and multi-class classification problems.

7.1 Binary Classification

We first introduce the experimental settings in Section 7.1.1, and then in Section 7.1.2 and Section 7.1.3, we compare our methods with SVM, SVR, LSSVM and three state-of-the-art approaches reviewed in Section 6. We study the margin distribution produced by ODM^L, ODM and SVM in Section 7.1.4. The computational cost is presented in Section 7.1.5.

7.1.1 Experimental Setup

We evaluate the effectiveness of our proposed methods on forty four regular scale data sets and ten large scale data sets. Table 1 summarizes the statistics of these data sets. The data set size is ranged from 44 to 2,396,130, and the dimension is ranged from 2 to 3,231,961, covering a broad range of properties. All features are normalized into the interval $[0, 1]$. For each data set, eighty percent of the instances are randomly selected as training data, and the rest are used as testing data. For regular scale data sets, both linear and RBF kernels are evaluated. ODM^L and ODM are trained by ODM$_{dcd}$ solver. Experiments are repeated for 30 times with random data partitions. For large scale data sets, only linear kernel is evaluated. ODM^L and ODM are trained using ODM$_{svrg}$ solver. Experiments are repeated for 10 times with random data partitions. The average accuracies as well as the standard deviations are recorded.

ODM^L and ODM are compared with standard SVM which ignores the margin distribution, SVR with binary targets $\{1, -1\}$, least squares SVM (LSSVM) and three state-of-the-art methods, that is, MDO, MAMC and KMOMD. For all the methods, the regularization parameter C or λ is selected

TABLE 1
Characteristics of fifty four binary data sets.

Scale	ID	Data set	#Instance	#Feature	ID	Data set	#Instance	#Feature
Regular	1	*duke*	44	7,129	23	*liver-disorders*	345	6
	2	*echocardiogram*	62	8	24	*house-votes*	435	16
	3	*colon-cancer*	62	2,000	25	*vehicle*	435	16
	4	*dbworld*	64	4,702	26	*wdbc*	569	14
	5	*leukemia*	72	7,129	27	*isolet*	600	51
	6	*balloons*	76	4	28	*credit-a*	653	15
	7	*hepatitis*	80	19	29	*breastw*	683	9
	8	*promoters*	106	57	30	*austra*	690	15
	9	*colic*	188	13	31	*australian*	690	42
	10	*parkinsons*	195	22	32	*diabetes*	768	8
	11	*colic.ORIG*	205	17	33	*fourclass*	862	2
	12	*sonar*	208	60	34	*tic-tac-toe*	958	9
	13	*house*	232	16	35	*credit-g*	1,000	20
	14	*vote*	232	16	36	*german*	1,000	59
	15	*heart-h*	261	10	37	*optdigits*	1,143	42
	16	*heart*	270	9	38	*svmguide3*	1,284	22
	17	*heart-statlog*	270	13	39	*madelon*	2,600	500
	18	*breast*	277	9	40	*splice*	3,175	60
	19	*cylinder-bands*	277	39	41	*farm-ads*	4,143	54,877
	20	*heart-c*	296	13	42	*spambase*	4,601	57
	21	*haberman*	306	14	43	*gisette*	7,000	5,000
	22	*vertebral-column*	310	6	44	*svmguide1*	7,089	4
Large	1	*phishing*	11,055	68	6	*cod-rna*	59,535	8
	2	*magic04*	19,020	10	7	*real-sim*	72,309	20,958
	3	*news20*	19,996	1,355,191	8	*mini-boo-ne*	130,064	50
	4	*adult-a*	32,561	123	9	*rcv1*	697,641	47,236
	5	*w8a*	49,749	300	10	*url*	2,396,130	3,231,961

by 5-fold cross validation from the set of $\{2^0, 2^2, \ldots, 2^{20}\}$ and the width of the RBF kernel is selected by 5-fold cross validation from the set of $\{2^{-4}\delta, 2^{-2}\delta, 2^0\delta, 2^2\delta, 2^4\delta\}$, where δ is the reciprocal of dimension. For MDO, the parameters are set as the recommended parameters in [36]. For ODM^L, the regularization parameters μ_1, μ_2 are selected from the set of $\{2^{-4}, 2^{-2}, 2^0, 2^2, 2^4\}$. For ODM, the regularization parameter μ and θ are selected from the set of $\{0.2, 0.4, 0.6, 0.8\}$. The parameters η used for ODM_{svrg} are set as the same setup in [29]. All selections are performed on training data.

7.1.2 Results on Regular Scale Data Sets

Tables 2 and 3 summarize the detailed results on the forty four regular scale data sets. As can be seen, the overall performance of our methods are superior or highly competitive to the other compared methods. Specifically, for linear kernel, ODM performs significantly better than SVM/SVR/LSSVM/MDO/MAMC/KMOMD/ODM^L on 31/34/35/35/41/31/19 over 44 data sets respectively, and achieves the best accuracy on 29 data sets; for RBF kernel, ODM performs significantly better than SVM/SVR/LSSVM/MAMC/KMOMD/ODM^L on 34/36/31/41/41/13 over 44 data sets respectively, and achieves the best accuracy on 30 data sets. In addition, as can be seen, in comparing with other methods which does not consider margin distribution, the win/tie/loss counts show that ODM are always better or comparable, almost never worse than it.

7.1.3 Results on Large Scale Data Sets

Table 4 summarizes the results on the ten large scale data sets. LSSVM and KMOMD did not return results on some data sets due to it lacks the mature packages to handle huge data. As can be seen, the overall performance of our methods are superior or highly competitive to other methods. Specifically, ODM performs significantly better than SVM/SVR/LSSVM/MDO/MAMC/KMOMD/ODM^L on 9/9/2/7/10/2/6 over 10 data sets, respectively, and achieves the best accuracy on 8 data sets.

7.1.4 Margin Distributions

Figure 2 plots the cumulative frequency of margin distribution for SVM, ODM^L and ODM on some representative regular scale data sets. As can be seen, the curves of ODM^L and ODM generally lies on the right side. If we fix a margin value $t > 0$, the proportion of instances with margin less than t for SVM is larger than our methods, i.e., our methods can make more instances with margin larger than t, which shows that the margin distribution of ODM^L and ODM are generally better than that of SVM. In other words, for most instances, our methods generally produce a larger margin than SVM.

7.1.5 Time Cost

We compare the one partition training time cost of our methods with SVM on the ten large scale data sets. All the experiments are performed with MATLAB 2012b on a machine with 8×2.60 GHz CPUs and 32GB main memory. The average CPU time (in seconds) on each data set is shown in Figure 3. We denote SVM implemented by the LIBLINEAR [32] package as SVM$_l$ and SVM implemented by SGD as SVM$_s$, respectively. It can be seen that, both SVM$_s$ and our methods are faster than SVM$_l$, owing to the use of SGD. ODM^L and ODM are just slightly slower

TABLE 2
Accuracy (mean±std.) comparison on regular scale data sets. Linear kernel is used. The best accuracy on each data set is bolded. •/○ indicates the performance of ODM is significantly better/worse than compared method (paired t-tests at 95% significance level). Average accuracy, top1 times and win/tie/loss counts for ODM are summarized in the last three rows.

Data set	SVM	SVR	LSSVM	MDO	MAMC	KMOMD	ODM^L	ODM
duke	84.2±8.8	84.2±8.8	85.0±8.2	80.6±8.7•	46.2±9.1•	84.1±8.8	85.0±8.0	**85.5±7.9**
echocardio	68.5±6.5•	71.7±7.1	70.2±6.4•	70.1±6.8•	71.0±5.7	72.0±6.6	67.3±7.7•	**73.0±6.7**
colon-canc	81.3±7.7	82.3±6.3	81.8±5.9•	80.3±7.0•	63.3±6.3•	81.0±7.0•	82.2±5.7	**82.3±5.5**
dbworld	86.5±5.2•	86.5±5.2•	86.5±5.2•	82.6±14.3•	78.0±19.4•	86.5±5.2•	88.2±3.6	**88.5±3.9**
leukemia	93.9±6.1•	93.3±6.3•	93.1±6.4•	93.5±6.4•	65.9±6.1•	93.9±6.1•	93.9±6.1•	**94.8±6.0**
balloons	68.1±5.9•	68.1±7.9•	71.8±5.7•	70.9±6.6•	60.9±8.3•	72.5±5.6•	72.4±6.7•	**73.9±5.6**
hepatitis	82.6±3.1•	82.6±3.1•	82.6±3.1•	83.5±3.8	82.6±3.1•	82.6±3.9•	83.9±3.8•	**84.8±4.6**
promoters	71.3±6.7•	71.4±6.4•	71.3±6.9•	69.6±7.5•	52.0±9.6•	71.9±6.6	**73.8±6.5**	73.7±6.4
colic	83.5±2.6	79.6±9.1•	82.4±5.6•	82.5±2.9	66.1±6.2•	80.7±3.2•	**83.8±2.8**	83.7±3.5
parkinsons	85.5±2.9•	76.4±3.5•	84.2±4.8•	86.0±2.7	76.4±3.5•	81.3±2.3•	**87.0±2.6**○	86.7±2.7
colic.ORIG	61.6±3.1•	60.6±3.7•	62.6±3.5	61.8±4.1•	62.3±2.7	63.5±4.6	63.0±4.6•	**63.8±3.9**
sonar	76.1±3.0	50.8±4.4•	72.0±3.9•	74.3±3.6	53.3±4.5•	**76.3±2.9**○	75.0±3.6	75.4±3.8
house	93.1±2.7•	**96.8±1.1**	96.4±1.3•	93.4±2.7•	88.3±2.9•	94.7±1.4•	96.8±1.1	96.8±1.1
vote	93.0±2.2•	**97.0±1.3**	97.0±1.3	93.4±2.4•	88.4±2.3•	94.9±2.0•	97.0±1.3	97.0±1.3
heart-h	79.6±2.9•	79.7±3.0•	79.7±3.2•	80.1±3.0•	63.5±4.1•	81.1±2.6	80.9±2.8	**81.3±2.4**
heart	82.6±2.7•	82.6±3.5	78.2±5.6•	83.1±2.8	53.7±5.7•	82.8±2.3•	83.4±2.3	**83.7±2.3**
heart-stat	83.2±2.4	82.7±2.6•	83.3±2.3	83.3±2.6	61.3±13.2•	81.7±2.9•	**84.0±2.5**	83.8±2.7
breast	71.3±2.8•	70.6±2.7•	72.6±2.8	71.7±3.0•	70.6±2.7•	69.6±3.1•	72.8±2.5•	**73.1±3.2**
cylinder-b	64.4±2.9•	65.8±3.8•	67.9±3.6•	70.2±3.2	64.4±2.9•	69.5±3.7•	70.7±3.3•	**70.8±3.3**
heart-c	79.7±2.3•	78.4±4.0•	79.8±2.5•	79.9±2.7•	65.9±11.7•	80.3±2.7	80.3±2.5	**80.5±2.4**
haberman	74.2±2.1	74.2±2.1	73.4±2.8	71.0±3.8•	73.8±2.0•	67.0±4.0•	73.3±3.1•	**74.3±2.2**
vertebral-	85.6±1.8	67.9±2.4•	77.2±3.4•	71.5±5.6•	67.9±2.4•	84.0±2.4•	85.5±1.9	**85.7±1.9**
liver-diso	67.3±2.8	62.2±4.7•	65.9±3.3•	57.0±6.1•	58.4±2.8•	63.6±3.4•	**67.8±3.0**○	67.7±3.9
house-vote	94.0±1.2•	94.7±1.0	91.8±1.6•	93.3±1.7•	78.9±6.0•	94.3±1.0•	94.5±0.8•	**94.7±0.9**
vehicle	95.5±1.2•	94.0±1.3•	95.9±1.1•	94.9±2.0•	56.6±16.0•	95.7±1.1•	96.0±1.0•	**96.2±1.1**
wdbc	95.9±1.5•	92.7±1.9•	94.7±1.6•	94.5±1.5•	62.3±2.0•	96.8±0.9	96.8±1.0•	**96.9±1.0**
isolet	99.4±0.4•	97.8±1.3•	99.3±0.4•	99.7±0.3•	62.1±20.7•	99.5±0.3•	99.7±0.3•	**99.8±0.3**
credit-a	86.4±1.2•	86.4±1.2•	86.4±1.2•	86.2±1.3•	59.6±6.3•	86.2±1.2•	86.4±1.2•	**86.5±1.2**
breastw	96.7±0.7	95.4±0.8•	95.3±0.9•	90.6±2.9•	35.0±2.1•	96.6±0.8	**96.9±0.8**○	96.8±0.7
austra	85.6±1.4•	85.6±1.3•	85.7±1.4•	85.5±1.4•	56.7±4.4•	85.8±1.3•	86.0±1.4•	**86.4±1.5**
australian	85.8±1.5•	86.1±1.7•	86.6±1.5•	84.8±1.4•	57.6±4.9•	86.2±1.6•	86.0±1.4•	**86.7±1.5**
diabetes	65.6±1.9•	65.6±1.9•	77.2±1.4	76.7±1.6•	65.6±1.9•	76.1±1.7•	77.4±1.6•	**77.4±1.7**
fourclass	76.7±1.9•	71.5±5.7•	76.6±1.2•	76.6±1.3•	64.1±2.0•	73.6±1.4•	**77.9±1.0**○	77.7±1.1
tic-tac-to	64.9±1.2•	64.9±1.2•	64.9±1.2•	63.9±5.5•	64.9±1.2•	57.9±2.1•	65.0±1.8•	**66.4±1.6**
credit-g	75.1±1.6	70.8±1.9•	74.8±1.6•	74.9±1.5	70.0±1.2•	73.3±1.8•	74.9±1.6•	**75.2±1.6**
german	73.7±1.4•	70.8±2.2•	73.9±1.4•	74.1±1.5	69.7±1.7•	72.9±1.7•	74.3±1.4	**74.4±1.2**
optdigits	99.7±0.2•	99.6±0.2•	99.7±0.2•	99.7±0.2•	74.3±20.4•	99.7±0.2•	99.8±0.2•	**99.9±0.2**
svmguide3	73.7±1.3•	78.5±1.8•	73.9±1.4•	79.0±2.0•	73.7±1.3•	78.8±1.6•	**82.0±1.5**○	81.6±1.3
madelon	58.0±2.4•	58.6±1.6	57.9±2.4•	56.8±1.0•	49.3±1.6•	55.9±1.2•	58.6±1.3	**58.8±1.4**
splice	84.1±0.7	83.5±0.7•	84.1±0.7	84.0±0.7•	58.9±10.0•	84.1±0.6	84.1±0.7	**84.2±0.7**
farm-ads	88.8±0.6•	88.3±0.7•	89.1±0.6•	88.6±0.7•	89.2±0.6	89.2±0.6	**89.4±0.8**	89.2±0.6
spambase	**92.7±0.5**○	85.3±0.8•	88.7±0.8•	89.3±0.6•	76.2±2.0•	92.3±0.6	92.3±0.6	92.3±0.7
gisette	96.8±0.3•	97.0±0.2	**97.5±0.2**	96.8±0.3•	74.4±5.6•	96.8±0.2•	97.4±0.3	97.5±0.3
svmguide1	**95.6±0.2**	87.9±0.8•	90.4±0.6•	88.9±0.4•	56.3±0.6•	95.3±0.3	95.5±0.3	95.5±0.2
Avg. Acc.	81.9	80.0	81.8	81.1	65.4	81.9	83.2	83.5
Top1 Time	2	2	1	0	0	1	9	29
ODM: w/t/l	31/12/1	34/10/0	35/9/0	35/9/0	41/3/0	31/12/1	19/20/5	

than SVM$_s$ on 2 data sets (w8a and cod-rna) but highly competitive with SVM$_s$ on the rest data sets. Note that both SVM$_l$ and SVM$_s$ are very fast implementations of SVMs, so this shows that our methods are also computationally efficient.

7.2 Multi-class Classification

We first introduce the experimental settings and baselines in Section 7.2.1, and then in Section 7.2.2, we compare our method with four versions of multi-class SVMs described in Section 2.2. We study the influence of the class size on generalization performance and margin distribution in Section 7.2.3. Finally, the computational cost is presented in Section 7.2.4.

7.2.1 Experimental Setup

We evaluate the effectiveness of our proposed methods on twenty two data sets. Table 5(a) summarizes the statistics of these data sets. The data set size ranges from 150 to 581,012, and the dimension ranges from 4 to 62,061. Moreover, the class size ranges from 3 to 1,000, so these data sets cover a broad range of properties. All features are normalized into the interval $[0, 1]$. For each data set, eighty percent of the instances are randomly selected as training data, and the rest are used as testing data. Experiments are repeated for 10 times with random data partitions, and the average accuracies as well as the standard deviations are recorded.

mcODM is compared with four versions of multi-class SVMs, i.e., ovrSVM, ovoSVM, ecocSVM and mcSVM. For

TABLE 3
Accuracy (mean±std.) comparison on regular scale data sets. RBF kernel is used. The best accuracy on each data set is bolded. •/○ indicates the performance of ODM is significantly better/worse than compared method (paired t-tests at 95% significance level). Average accuracy, top1 times and win/tie/loss counts for ODM are summarized in the last three rows. MDO is not involved here since it is specified for the linear kernel.

Data set	SVM	SVR	LSSVM	MAMC	KMOMD	ODM^L	ODM
duke	84.2±8.2•	84.2±8.2•	81.7±9.7•	71.8±11.1•	74.4±14.7•	85.3±7.9	**85.6±8.2**
echocardio	61.1±7.3•	70.8±6.3•	73.2±6.5	71.0±5.7•	67.4±6.1•	74.0±6.0	**74.4±6.7**
colon-canc	80.9±10.9	78.4±7.7•	78.5±8.1•	63.7±7.2•	77.0±8.5•	**83.3±5.3**	82.7±6.3
dbworld	86.4±5.2•	86.4±5.2•	85.7±5.5•	58.6±12.1•	57.2±14.3•	87.2±5.9	**87.6±4.3**
leukemia	93.9±6.1	93.3±6.3•	92.9±6.4•	68.0±8.6•	86.6±10.0•	**95.0±5.5**	94.6±5.6
balloons	70.0±6.1•	66.6±7.1•	71.6±5.3•	72.1±5.6•	67.0±7.5•	73.2±4.9•	**75.1±4.5**
hepatitis	82.6±3.1•	83.0±3.5	83.0±3.3	82.6±3.1•	**84.2±4.3**	84.0±4.0	84.2±3.9
promoters	71.5±6.7•	72.1±6.5•	71.3±6.4•	66.6±11.1•	70.6±6.4•	73.8±6.5	**74.7±5.4**
colic	81.1±3.9•	**84.1±2.4**	82.6±3.0•	82.6±2.6•	82.6±3.6•	83.8±2.8	84.1±2.4
parkinsons	88.7±2.7•	90.0±3.1•	89.7±3.4•	92.5±2.8•	84.7±2.6•	89.2±2.4•	**93.5±2.8**
colic.ORIG	64.1±4.1	61.4±4.3•	63.5±3.5•	62.5±4.0•	62.6±4.0•	64.2±4.1	**65.2±3.8**
sonar	78.3±3.0•	55.2±7.4•	73.8±3.6•	50.7±4.3•	71.4±4.2•	85.7±3.6	**85.8±3.6**
house	95.9±1.7•	**96.8±1.1**	96.8±1.1	56.1±13.9•	93.4±2.7•	96.8±1.1	96.8±1.1
vote	95.7±1.6•	**97.0±1.3**	97.0±1.3	54.7±12.3•	93.7±3.0•	97.0±1.3	97.0±1.3
heart-h	80.4±2.8•	79.4±2.6•	80.7±2.6•	79.7±2.9•	80.4±3.2•	80.9±2.8	**81.3±2.3**
heart	82.6±2.7•	82.1±3.1•	83.7±2.3	79.9±5.1•	82.9±2.9•	83.4±2.3	**83.9±2.3**
heart-stat	82.8±2.9•	81.5±3.4•	83.0±2.7•	82.6±2.7•	80.8±3.5•	**84.0±2.5**	83.8±2.5
breast	72.9±2.9•	73.3±2.4•	75.0±2.9•	74.3±2.3•	70.6±2.6•	73.0±2.8•	**75.5±2.7**
cylinder-b	73.8±3.6•	74.1±3.8•	73.9±3.8•	69.8±3.3•	71.9±4.1•	74.4±3.6•	**75.4±3.0**
heart-c	79.6±2.9•	79.7±2.6•	80.2±2.9•	78.7±2.2•	79.2±2.5•	80.3±2.5	**80.9±2.7**
haberman	74.2±2.1	73.7±2.0•	72.0±3.1•	74.2±2.1	73.6±2.2	74.2±2.2	**74.2±2.2**
vertebral-	78.6±3.0•	78.2±2.8•	78.9±2.5•	77.3±3.0•	76.9±3.1•	**85.6±1.9**	85.5±2.1
liver-diso	71.8±3.2	69.2±3.5•	71.4±3.3•	61.3±3.9•	70.0±4.6•	72.1±3.2	**72.3±3.1**
house-vote	94.7±1.2•	93.6±1.1•	93.9±1.0•	60.6±2.3•	91.4±2.4•	94.8±1.2•	**95.1±1.2**
vehicle	98.8±0.8•	98.8±0.7•	99.1±0.7•	49.6±8.0•	98.8±0.8•	99.0±0.8•	**99.4±0.6**
wdbc	97.1±1.1	95.3±1.1•	96.4±1.2•	92.9±1.5•	94.5±2.4•	97.2±1.0	**97.4±1.0**
isolet	99.6±0.3•	99.8±0.3	99.6±0.5•	99.3±0.5•	99.6±0.3•	99.8±0.2	**99.9±0.2**
credit-a	86.4±1.2•	86.4±1.2•	86.4±1.2	86.1±1.5	83.9±2.8•	86.4±1.2•	**86.5±1.4**
breastw	96.7±0.9•	96.6±0.7•	96.8±0.7•	96.1±0.7•	**97.3±0.6**○	96.9±1.1	97.0±0.7
austra	85.7±1.3•	84.9±1.9•	85.2±2.0•	83.4±1.7•	85.1±1.8•	**86.3±1.4**	86.2±1.4
australian	85.8±1.5•	85.8±1.5•	86.3±1.7	85.1±1.5•	86.0±1.6•	86.0±1.4•	**86.8±1.3**
diabetes	77.4±1.4•	76.6±1.8•	77.0±1.6•	73.1±1.6•	73.8±3.4•	77.4±1.3•	**77.8±1.7**
fourclass	99.8±0.3•	99.8±0.3•	99.9±0.1•	99.9±0.1•	99.7±0.3•	99.9±0.1•	**100.0±0.0**
tic-tac-to	98.2±0.4	98.2±0.4	98.2±0.4	82.4±1.2•	78.6±2.4•	98.2±0.4	**98.4±0.6**
credit-g	75.3±1.8•	74.4±1.5•	75.5±1.5•	72.7±1.5•	74.8±2.0•	75.3±1.7•	**75.8±1.4**
german	74.2±1.5•	74.6±1.4•	75.0±1.2	72.0±1.6•	74.6±1.7•	74.3±1.3•	**75.1±1.2**
optdigits	99.9±0.2•	99.9±0.1•	99.9±0.1	49.1±0.8•	99.8±0.2•	99.9±0.2•	**99.9±0.1**
svmguide3	82.5±1.5•	81.0±1.7•	82.5±1.5•	78.2±1.2•	78.3±1.4•	82.7±1.4	**83.1±0.9**
madelon	50.4±3.3•	50.4±3.3•	59.5±1.6	**61.3±1.5**○	58.0±1.7•	59.7±1.2	59.9±1.2
splice	89.6±0.6•	89.7±0.6•	90.1±0.6•	71.5±9.7•	81.1±1.6•	90.4±0.6	**90.5±0.6**
farm-ads	89.1±0.8	89.0±0.7	**89.3±0.7**	81.6±1.2•	87.3±1.3•	89.3±1.1	89.3±1.2
spambase	93.2±0.5•	92.6±0.4•	93.0±0.5•	89.6±0.7•	92.3±0.7•	93.5±0.2	**93.6±0.4**
gisette	97.6±0.2	97.7±0.2	**97.8±0.2**	95.5±0.3•	88.5±7.2•	97.6±0.2	97.6±0.3
svmguide1	**97.0±0.2**	96.7±0.2•	96.9±0.2•	96.2±0.2•	95.7±0.8•	97.0±0.3	97.0±0.3
Avg. Acc.	84.1	83.5	84.5	75.2	81.3	85.5	86.0
Top1 Time	1	3	2	1	2	5	30
ODM: w/t/l	34/10/0	36/8/0	31/13/0	41/2/1	41/2/1	13/31/0	

all the methods, the regularization parameter λ for mcODM or C for binary SVM and mcSVM is selected by 5-fold cross validation from the set of $\{2^0, 2^2, \ldots, 2^{20}\}$. For mcODM, the regularization parameters μ and θ are both selected from the set of $\{0.2, 0.4, 0.6, 0.8\}$. For ecocSVM, the exhaustive codes strategy is employed, i.e., for each class, we construct a code of length $2^{k-1} - 1$ as the signature. All the selections for parameters are performed on training sets.

7.2.2 Results

Table 5(b) summarizes the detailed results on twenty two data sets. As can be seen, the overall performance of our method is superior or highly competitive to the other compared methods. Specifically, mcODM performs significantly better than mcSVM/ovrSVM/ovoSVM/ecocSVM on 17/19/18/17 over 22 data sets respectively, and achieves the best accuracy on 20 data sets.

7.2.3 Influence of the Class Size

Figure 4 plots the generalization performance of all the five methods on data set segment, and similar observation can be found for other data sets. As can be seen, when the class size is less than four, all methods perform quite well. However, as the fifth class is added, the generalization performance of the other four methods decreases drastically. This might be attributable to the introduction of some noisy data, which SVM-style algorithms are very sensitive to since they optimize the minimum margin. On the other hand, our method considers the whole margin distribution, so it can be robust to noise and behave more stably.

TABLE 4
Accuracy (mean±std.) comparison on large scale data sets. Linear kernel is used. The best accuracy on each data set is bolded. •/○ indicates the performance of ODM is significantly better/worse than compared method (paired t-tests at 95% significance level). Average accuracy, top1 times and win/tie/loss counts for ODM are summarized in the last three rows. LSSVM and KMOMD did not return results on some data sets in 48 hours.

Data set	SVM	SVR	LSSVM	MDO	MAMC	KMOMD	ODM^L	ODM
phishing	91.3±3.3•	84.8±14.0	92.9±0.4•	93.8±0.3•	55.6±0.6•	93.6±0.3•	93.8±0.3	**93.9±0.3**
magic04	79.2±0.3	77.0±0.8•	78.4±0.3•	**79.2±0.3**	64.9±0.3•	78.8±0.4•	79.0±0.3	79.2±0.3
news20	95.4±0.1•	95.4±0.1•	95.6±0.2	94.9±0.3•	77.2±1.7•	95.5±0.2	95.5±0.2•	**95.6±0.2**
adult-a	84.5±0.2•	80.5±0.5•	N/A	84.5±0.2•	75.9±0.2•	N/A	84.6±0.2	**84.6±0.2**
w8a	97.1±0.1•	89.0±0.2•	N/A	98.6±0.1	88.9±0.2•	N/A	98.6±0.1	**98.6±0.0**
cod-rna	89.9±0.1•	88.6±1.0•	N/A	**93.8±0.1**○	66.7±0.1•	N/A	93.7±0.1•	93.8±0.1
real-sim	96.2±0.1•	96.1±0.1•	N/A	96.8±0.1•	74.2±0.4•	N/A	97.2±0.1•	**97.2±0.1**
mini-boo-ne	86.1±0.6•	81.7±2.2•	N/A	84.6±0.1•	72.0±0.1•	N/A	85.3±0.1•	**90.6±0.1**
rcv1	97.7±0.0•	97.1±0.0•	N/A	97.7±0.0•	91.3±0.0•	N/A	97.8±0.0•	**97.8±0.0**
url	99.3±0.0•	99.2±0.0•	N/A	99.3±0.0•	67.0±0.0•	N/A	99.1±0.0•	**99.3±0.0**
Avg. Acc.	91.7	89.0	N/A	92.3	73.4	N/A	92.5	93.1
Top1 Time	0	0	0	2	0	0	0	8
ODM: w/t/l	9/1/0	9/1/0	2/1/0	7/2/1	10/0/0	2/1/0	6/4/0	

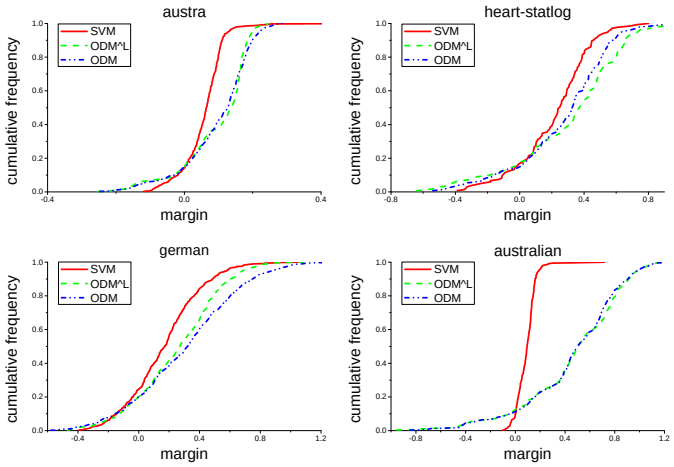

Fig. 2. Cumulative frequency (y-axis) with respect to margin (x-axis) of SVM, ODM^L and ODM on some representative regular scale data sets. Consider a straight line parallel to y-axis with parametric equation as $x = t$ and intersect with the curves, then the y-axis coordinate of the intersection points is the proportion of instances which have margin less than t.

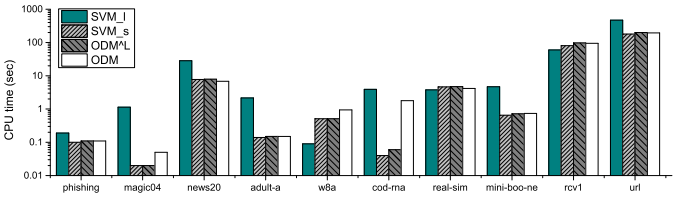

Fig. 3. One partition training time cost of SVM$_l$, SVM$_s$, ODM^L and ODM on the ten large scale data sets.

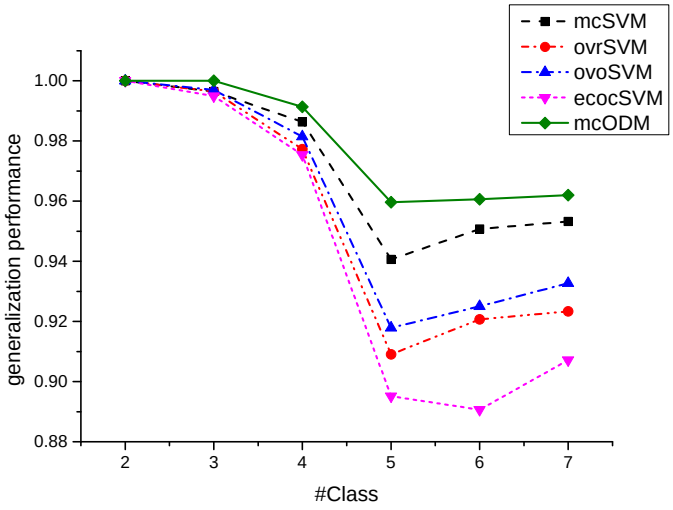

Fig. 4. Generalization performance of all the five methods on data set segment with the increase of the number of classes.

7.2.4 Time Cost

We compare the one partition training time cost of our method with mcSVM, ovrSVM, ovoSVM on all the data sets except aloi, on which ovoSVM could not return results in 48 hours. All the experiments are performed with MATLAB 2012b on a machine with 8×2.60 GHz CPUs and 32GB main memory. The average CPU time (in seconds) on each data set is shown in Figure 5. The binary SVM used in ovrSVM, ovoSVM and mcSVM are both implemented by the LIBLINEAR [32] package. It can be seen that for small data sets, the efficiency of all the methods are similar, however, for data sets with more than ten classes, e.g., sector and rcv1, mcSVM and mcODM, which learn all the scroing functions at once, are much faster than ovrSVM and ovoSVM, owing to the inefficiency of binary-decomposition as discussed in Section 2.2. Note that LIBLINEAR are very fast implementations of binary SVM and mcSVM, so this shows that our method is also computationally efficient.

8 Conclusions

Support vector machine works by maximizing the minimum margin. However, theoretical results suggested that the margin distribution, rather than the single-point margin, is more crucial to the generalization performance. In this paper, we propose the Optimal margin Distribution Machine (ODM), which tries to optimize the margin distribution by

TABLE 5
(a) Characteristics of twenty two multi-class data sets. (b) Accuracy (mean±std.) comparison. Linear kernel is used. The best accuracy on each data set is bolded. ●/○ indicates the performance of mcODM is significantly better/worse than the compared methods (paired t-tests at 95% significance level). Average rank, top1 times and win/tie/loss counts for mcODM are listed in the last three rows. ovoSVM and ecocSVM did not return results on some data sets in 48 hours.

(a)

ID	Data set	#Instance	#Feature	#Class
1	*iris*	150	4	3
2	*wine*	178	13	3
3	*glass*	214	9	6
4	*svmguide2*	391	20	3
5	*svmguide4*	612	10	6
6	*vehicle*	846	18	4
7	*vowel*	990	10	11
8	*segment*	2,310	19	7
9	*dna*	3,186	180	3
10	*satimage*	6,435	36	6
11	*usps*	9,298	256	10
12	*sector*	9,619	55,197	105
13	*pendigits*	10,992	16	10
14	*news20*	19,928	62,061	20
15	*letter*	20,000	16	26
16	*protein*	24,387	357	3
17	*shuttle*	58,000	9	7
18	*connect-4*	67,557	126	3
19	*mnist*	70,000	780	10
20	*aloi*	108,000	128	1,000
21	*rcv1*	534,135	47,236	53
22	*covtype*	581,012	54	7

(b)

Data set	mcSVM	ovrSVM	ovoSVM	ecocSVM	mcODM
iris	84.7±5.0●	82.0±5.0●	73.0±7.1●	73.3±4.2●	**87.3±3.4**
wine	96.5±2.6●	97.0±2.4	96.8±3.1	91.6±2.7●	**98.4±1.9**
glass	62.0±7.6●	57.8±2.1●	62.4±6.0●	60.4±4.0●	**68.2±7.3**
svmguide2	80.9±2.6●	81.5±2.3●	74.6±5.4●	79.0±3.6●	**84.1±1.8**
svmguide4	86.6±3.2●	77.2±3.1●	77.1±6.9●	74.2±4.3●	**87.8±3.5**
vehicle	80.5±2.1●	78.5±2.9●	73.5±3.5●	76.9±2.5●	**85.6±6.6**
vowel	65.4±2.1	44.1±2.6●	**71.1±4.9**	43.1±2.0●	65.6±3.9
segment	95.3±0.9●	92.3±0.7●	93.2±2.2●	90.7±0.8●	**96.7±1.3**
dna	95.1±0.6●	95.1±0.7●	93.0±1.0●	90.9±1.1●	**95.6±0.7**
satimage	81.8±0.5●	80.1±0.7●	77.4±4.2●	76.0±1.4●	**89.1±8.0**
usps	**95.1±0.3**	94.2±0.3●	94.5±0.6●	92.3±0.7●	95.1±0.6
sector	93.3±0.4●	93.1±0.5●	89.9±0.6●	N/A	**93.6±0.6**
pendigits	94.8±0.3●	91.8±0.5●	95.7±0.8●	87.6±0.7●	**96.7±0.9**
news20	83.3±0.3●	83.0±0.3●	69.4±0.4●	N/A	**83.6±0.3**
letter	76.9±0.4●	64.9±1.0●	75.7±1.2●	N/A	**82.7±1.6**
protein	68.7±0.3	68.7±0.4	68.1±0.3●	66.5±0.5●	**70.2±2.3**
shuttle	97.2±0.1●	89.9±2.8●	97.4±0.1●	84.2±0.0●	**99.6±0.0**
connect-4	75.7±0.1●	75.7±0.1●	75.7±0.2●	75.1±0.2●	**94.4±7.6**
mnist	92.5±0.2●	91.5±0.2●	90.9±0.5●	87.6±0.2●	**95.2±0.3**
aloi	90.1±0.2●	82.2±0.2●	N/A	N/A	**91.4±0.3**
rcv1	92.8±0.1	92.7±0.1	92.1±0.1●	N/A	**93.0±1.1**
covtype	71.7±0.1	71.2±0.1●	72.7±0.1	70.6±0.1●	**72.8±1.6**
Avg. Rank	2.45	3.18	3.45	4.82	1.09
Top1 Time	3	0	4	0	15
mcODM: w/t/l	17/5/0	19/3/0	18/3/0	17/0/0	

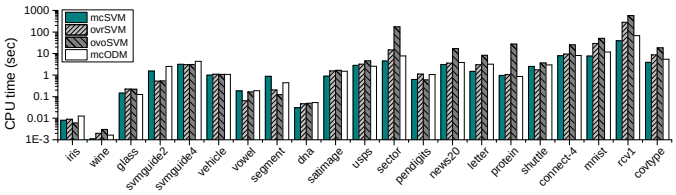

Fig. 5. Full training time cost of mcSVM, ovrSVM, ovoSVM and mcODM on all the data sets except aloi.

considering the margin mean and variance simultaneously. Our method is a general learning approach which can be applied in any place where SVMs are used. Extensive experiments on fifty four binary data sets and twenty two multi-class data sets validate the superiority of our method to SVM in comparisons with other state-of-the-art methods. In the future it will be interesting to generalize the idea of ODM to other learning settings, i.e., multi-label learning [40] and more cases of weakly supervised learning [14], and try some recent nonconvex optimization algorithms such as [41] for solving mcODM.

ACKNOWLEDGMENT

This research was supported by the National Key R&D Program of China (2018YFB1004300), National Science Foundation of China (61751306), and the Collaborative Innovation Center of Novel Software Technology and Industrialization.

REFERENCES

[1] C. Cortes and V. N. Vapnik, "Support-vector networks," *Machine Learning*, vol. 20, no. 3, pp. 273–297, 1995.

[2] C.-P. Lee and D. Roth, "Distributed box-constrained quadratic optimization for dual linear svm," in *Proceedings of the 32nd International Conference on Machine Learning*, Lille, France, 2015, pp. 987–996.

[3] A. Osokin, J. B. Alayrac, I. Lukasewitz, P. K. Dokania, and S. Lacoste-Julien, "Minding the gaps for block frank-wolfe optimization of structured svms," in *Proceedings of the 33rd International Conference on Machine Learning*, New York, NY, 2016, pp. 593–602.

[4] W. Zhang, B. Hong, W. Liu, J. Ye, D. Cai, X. He, and J. Wang, "Scaling up sparse support vector machines by simultaneous feature and sample reduction," in *Proceedings of the 34th International Conference on Machine Learning*, Sydney, Australia, 2017, pp. 4016–4025.

[5] V. N. Vapnik, *The Nature of Statistical Learning Theory*. New York, NY: Springer, 1995.

[6] Y. Freund and R. E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," in *Proceedings of the 2nd European Conference on Computational Learning Theory*, Barcelona, Spain, 1995, pp. 23–37.

[7] Z.-H. Zhou, *Ensemble Methods: Foundations and Algorithms*. Boca Raton, FL: CRC Press, 2012.

[8] R. E. Schapire, Y. Freund, P. Barlett, and W. S. Lee, "Boosting the margin: a new explanation for the effectives of voting methods," *Annuals of Statistics*, vol. 26, no. 5, pp. 1651–1686, 1998.

[9] L. Breiman, "Prediction games and arcing algorithms," *Neural Computation*, vol. 11, no. 7, pp. 1493–1517, 1999.

[10] L. Reyzin and R. E. Schapire, "How boosting the margin can also boost classifier complexity," in *Proceedings of 23rd International Conference on Machine Learning*, Pittsburgh, PA, 2006, pp. 753–760.

[11] W. Gao and Z.-H. Zhou, "On the doubt about margin explanation of boosting," *Artificial Intelligence*, vol. 203, pp. 1–18, 2013.

[12] T. Zhang and Z.-H. Zhou, "Large margin distribution machine," in *Proceedings of the 20th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, New York, NY, 2014, pp. 313–322.

[13] Y.-H. Zhou and Z.-H. Zhou, "Large margin distribution learning with cost interval and unlabeled data," *IEEE Transactions on Knowledge and Data Engineering*, vol. 28, no. 7, pp. 1749–1763, 2016.

[14] Z.-H. Zhou, "A brief introduction to weakly supervised learning," *National Science Review*, vol. 5, no. 1, pp. 44–53, 2018.

[15] T. Zhang and Z.-H. Zhou, "Optimal margin distribution clustering," in *Proceedings of the 32nd AAAI Conference on Artificial Intelligence*, New Orleans, LA, 2018, pp. 4474–4481.

[16] G. Ou, Y. Wang, W. Pang, and G. M. Coghill, "Large margin distribution machine recursive feature elimination," in *The 4th International Conference on Systems and Informatics*, Hangzhou, China, 2017, pp. 1518–1523.

[17] Z.-H. Zhou, "Large margin distribution learning," in *Proceedings of the 6th IAPR International Workshop on Artificial Neural Networks in Pattern Recognition*, Montreal, Canada, 2014, pp. 1–11.

[18] H. G. K. Ulrich, "Pairwise classification and support vector machines," in *Advances in Kernel Methods: Support Vector Machines*. Cambridge, MA: MIT Press, 1998, pp. 255–268.

[19] T. G. Dietterich and G. Bakiri, "Solving multiclass learning problems via error-correcting output codes," *Journal of Artificial Intelligence Research*, vol. 2, no. 1, pp. 263–286, 1995.

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

[21] F. Nie, X. Wang, and H. Huang, "Multiclass capped l_p-norm SVM for robust classifications," in *Proceedings of the 31st AAAI Conference on Artificial Intelligence*, San Francisco, CA, 2017, pp. 2415–2421.

[22] J. Xu, X. Liu, Z. Huo, C. Deng, F. Nie, and H. Huang, "Multi-class support vector machine via maximizing multi-class margins," in *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, Melbourne, Australia, 2017, pp. 3154–3160.

[23] B. Schölkopf and A. J. Smola, *Learning with kernels: support vector machines, regularization, optimization, and beyond*. Cambridge, MA: MIT Press, 2001.

[24] G. X. Yuan, C. H. Ho, and C.-J. Lin, "Recent advances of large-scale linear classification," *Proceedings of the IEEE*, vol. 100, no. 9, pp. 2584–2603, 2012.

[25] C. J. Hsieh, K. W. Chang, C.-J. Lin, S. S. Keerthi, and S. Sundararajan, "A dual coordinate descent method for large-scale linear svm," in *Proceedings of the 25th International Conference on Machine Learning*, Helsinki, Finland, 2008, pp. 408–415.

[26] T. Zhang, "Solving large scale linear prediction problems using stochastic gradient descent algorithms," in *Proceedings of the 21st International Conference on Machine Learning*, Banff, Canada, 2004, pp. 116–123.

[27] H. Kushner and G. G. Yin, *Stochastic approximation and recursive algorithms and applications*. New York, NY: Springer, 2003.

[28] L. Bottou, "Large-scale machine learning with stochastic gradient descent," in *Proceedings of the 19th International Conference on Computational Statistics*, Paris, France, 2010, pp. 177–186.

[29] R. Johnson and T. Zhang, "Accelerating stochastic gradient descent using predictive variance reduction," in *Advances in Neural Information Processing Systems 26*, Lake Tahoe, NV, 2013, pp. 315–323.

[30] T. Zhang and Z.-H. Zhou, "Multi-class optimal distribution machine," in *Proceedings of the 34th International Conference on Machine Learning*, Sydney, Australia, 2017, pp. 4063–4071.

[31] P. Tseng, "Convergence of a block coordinate descent method for nondifferentiable minimization," *Journal of Optimization Theory and Applications*, vol. 109, no. 3, pp. 475–494, 2001.

[32] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin, "LIBLINEAR: A library for large linear classification," *Journal of Machine Learning Research*, vol. 9, pp. 1871–1874, 2008.

[33] C. McDiarmid, "On the method of bounded differences," *Surveys in Combinatorics*, vol. 141, no. 1, pp. 148–188, 1989.

[34] M. Mohri, A. Talwalkar, and A. Rostamizadeh, *Foundations of Machine Learning*. Cambridge, MA: MIT Press, 2012.

[35] Y. Lei, Ürün Dogan, A. Binder, and M. Kloft, "Multi-class svms: From tighter data-dependent generalization bounds to novel algorithms," in *Advances in Neural Information Processing Systems 28*, Montreal, Canada, 2015, pp. 2026–2034.

[36] A. Garg and D. Roth, "Margin distribution and learning algorithms," in *Proceedings of the 20th International Conference on Machine Learning*, Washington, DC, 2003, pp. 210–217.

[37] K. Pelckmans, J. A. K. Suykens, and B. De Moor, "A risk minimization principle for a class of parzen estimators," in *Advances in Neural Information Processing Systems 20*, Vancouver, Canada, 2007, pp. 1137–1144.

[38] F. Aiolli, G. S. Martino, and A. Sperduti, "A kernel method for the optimization of the margin distribution," in *Proceedings of the 18th International Conference on Artificial Neural Networks*, Prague, Czech, 2008, pp. 305–314.

[39] Y. Freund and R. E. Schapire, "Large margin classification using the perceptron algorithm," *Machine Learning*, vol. 37, pp. 277–296, 1999.

[40] M.-L. Zhang, Y.-K. Li, X.-Y. Liu, and X. Geng, "Binary relevance for multi-label learning: an overview," *Frontiers of Computer Science*, vol. 12, no. 2, pp. 191–202, 2018.

[41] F. Wang, W. Cao, and Z. Xu, "Convergence of multi-block bregman admm for nonconvex composite problems," *Science China Information Sciences*, vol. 61, no. 12, pp. 122 101:1–122 101:12, 2018.

Teng Zhang received the BSc degree in Computer Science and Technology from Nanjing University, Nanjing, China, in 2010. In the same year, he was admitted to further study in Nanjing University, Nanjing, China. Now, he is currently a PhD candidate and a member of LAMDA Group. His research interests mainly include margin based methods and optimization.

Zhi-Hua Zhou (S'00-M'01-SM'06-F'13) received the BSc, MSc and PhD degrees in computer science from Nanjing University, China, in 1996, 1998 and 2000, respectively, all with the highest honors. He joined the Department of Computer Science & Technology at Nanjing University as an Assistant Professor in 2001, and is currently Professor, Head of the Department of Computer Science and Technology, and Dean of the School of Artificial Intelligence; he is also the Founding Director of the LAMDA group. His research interests are mainly in artificial intelligence, machine learning and data mining. He has authored the books *Ensemble Methods: Foundations and Algorithms* and *Machine Learning* (in Chinese), and published more than 150 papers in top-tier international journals or conference proceedings. He has received various awards/honors including the National Natural Science Award of China, the PAKDD Distinguished Contribution Award, the IEEE ICDM Outstanding Service Award, the Microsoft Professorship Award, etc. He also holds 24 patents. He is the Editor-in-Chief of the *Frontiers of Computer Science*, Associate Editor-in-Chief of the *Science China Information Sciences*, Action or Associate Editor of the *Machine Learning*, *IEEE Transactions on Pattern Analysis and Machine Intelligence* , *ACM Transactions on Knowledge Discovery from Data* , etc. He served as Associate Editor-in-Chief for *Chinese Science Bulletin* (2008-2014), Associate Editor for *IEEE Transactions on Knowledge and Data Engineering* (2008-2012), *IEEE Transactions on Neural Networks and Learning Systems* (2014-2017), *ACM Transactions on Intelligent Systems and Technology* (2009-2017), Neural Networks (2014-2016), etc. He founded ACML (Asian Conference on Machine Learning), served as Advisory Committee member for IJCAI (2015-2016), Steering Committee member for ICDM, PAKDD and PRICAI, and Chair of various conferences such as General co-chair of ICDM 2016 and PAKDD 2014, Program co-chair of AAAI 2019 and SDM 2013, and Area chair of NIPS, ICML, AAAI, IJCAI, KDD, etc. He is/was the Chair of the IEEE CIS Data Mining Technical Committee (2015-2016), the Chair of the CCF-AI (2012-), and the Chair of the CAAI Machine Learning Technical Committee (2006-2015). He is a foreign member of the Academy of Europe, and a Fellow of the ACM, AAAI, AAAS, IEEE, IAPR, IET/IEE, CCF, and CAAI.