Scalable Optimal Margin Distribution Machine

Yilin Wang, **Nan Cao**, **Teng Zhang***, **Xuanhua Shi** and **Hai Jin**
National Engineering Research Center for Big Data Technology and System
Services Computing Technology and System Lab, Cluster and Grid Computing Lab
School of Computer Science and Technology, Huazhong University of Science and Technology, China
{yilin_wang, nan_cao, tengzhang, xhshi, hjin}@hust.edu.cn

Abstract

Optimal margin Distribution Machine (ODM) is a newly proposed statistical learning framework rooting in the latest margin theory, which demonstrates better generalization performance than the traditional large margin based counterparts. However, it suffers from the ubiquitous scalability problem regarding both computation time and memory storage as other kernel methods. This paper proposes a scalable ODM, which can achieve nearly ten times speedup compared to the original ODM training method. For nonlinear kernels, we put forward a novel distribution-aware partition method to make the local ODM trained on each partition be close and converge fast to the global one. When linear kernel is applied, we extend a communication efficient SVRG method to accelerate the training further. Extensive empirical studies validate that our proposed method is highly computational efficient and almost never worsen the generalization.

1 Introduction

Recently, the study on margin theory [Gao and Zhou, 2013] demonstrates an upper bound disclosing that maximizing the minimum margin does not necessarily result in a good performance. Instead, the distribution rather than a single margin is much more critical. Later on, the study on lower bound [Grønlund *et al.*, 2019] further proves that the upper bound is almost optimal up to a logarithmic factor. Inspired by these insightful works, Zhang and Zhou [2019] propose the *Optimal margin Distribution Machine* (ODM), which explicitly optimizes the margin distribution by maximizing the mean and minimizing the variance simultaneously and exhibits much better generalization than the traditional large margin based counterparts. Due to the superiority shown on both binary and multi-class classification tasks, many works attempt to extend ODM to more genreal learning settings, just to list a few, cost-sensitive learning [Zhou and Zhou, 2016; Cheng *et al.*, 2017], weakly supervised learning [Zhang and Zhou, 2018a; Zhang and Zhou, 2018b; Luan *et al.*, 2020; Zhang and Jin, 2020; Cao *et al.*, 2022], multi-label learning [Tan *et al.*, 2020; Cao *et al.*, 2021], online learning [Zhang *et al.*, 2020], and regression [Rastogi *et al.*, 2020]. Plenty of successes on various learning tasks validate the superiority of this new statistical learning framework. However, with the dramatic progress of digital technologies, the data generated devices become as diverse as computers, mobile phones, smartwatches, cars, etc., and the amount of data created each day grows tremendously, thus these ODM based extensions suffer from the scalability problem regarding both computation time and memory storage as other kernel methods.

There have been many works devoted to accelerating kernel methods, which can be roughly classified into three categories. The first category is based on approximation, e.g., the random Fourier feature [Rahimi and Recht, 2007] takes the trigonometric functions as basis functions to approximate the kernel mapping, the Nyström method [Williams and Seeger, 2001] generates a low-rank approximations by sampling a subset of columns, and the coreset [Tan *et al.*, 2019] adaptively sketches the whole data by choosing some landmark points. The second category divides the data into partitions on which local models are trained and combined to produce a larger local or global model, e.g., in [Graf *et al.*, 2004; Hsieh *et al.*, 2014; Singh *et al.*, 2017], a tree architecture on partitions is designed first, guided by which the solutions of different partitions are aggregated; in [Yu *et al.*, 2005; Navia-Vazquez *et al.*, 2006; Loosli *et al.*, 2007], the key instances identification and exchange are further introduced to accelerate the training; in [Si *et al.*, 2017], both low-rank and clustering structure of the kernel matrix are taken into account to get an approximation of kernel matrix. The third category is directly applying the distributed-style optimization method, such as the augmented Lagrangian method [Forero *et al.*, 2010] and the alternating direction method of multipliers [Boyd *et al.*, 2010], or extending existing solver to a distributed environment, e.g., distributed SMO [Cao *et al.*, 2006].

Notice that the random Fourier feature adopts a data-independent kernel mapping and the Nyström method takes a data distribution-unaware sampling, hence their performance are both inferior to the coreset method [Tan *et al.*, 2019], which inspires us to leverage data as heavily as possible. Moreover, the distributed off-the-shelf *quadratic programming* (QP) solvers can be directly applied to train ODM, but

*Corresponding Author

they are all general approaches thus ignore the intrinsic structure of the problem and can hardly achieve the greatest efficiency. To take the best of both worlds, this paper proposes a specially designed *scalable ODM* (SODM). Specifically, we put forward a novel data partition method so that ODM trained on each partition has a solution close to that trained on the whole data. When some partitions are merged to form a larger partition, the solution on it can be quickly obtained by concatenating the previous local solutions as the initial point. Besides, in the case of the linear kernel, we extend a communication efficient SVRG method to accelerate the training further. To summarize, the remarkable differences of SODM compared with existing scalable QP solvers are threefold:

1. SODM incorporates a novel partition strategy, which makes the local ODM on each partition be close to the global one so that the training can be accelerated.
2. SODM accelerates the training further when the linear kernel is applied by extending a communication efficient SVRG.
3. SODM achieves nearly ten times speedup meanwhile, maintain ODM's generalization performance in most situations.

The rest of this paper is organized as follows. We first introduce some preliminaries, and then present the technical detail of our method. After that we show the experimental results and empirical observations. Finally we conclude the paper with future work.

2 Preliminaries

Throughout the paper, scalars are denoted by normal case letters (e.g., m and M). Vectors and matrices are denoted by boldface lower and upper case letters, respectively (e.g., $\boldsymbol{x}$ and $\mathbf{X}$). The (i,j)-th entry of matrix $\mathbf{X}$ is $[\mathbf{X}]_{ij}$. Sets are designated by upper case letters with mathcal font (e.g., $\mathcal{S}$). The input space is $\mathcal{X} \subseteq \mathbb{R}^N$ and $\mathcal{Y} = \{1, -1\}$ is the label set. For any positive integer M, the set of integers $\{1, \ldots, M\}$ is denoted by $[M]$. For the feature mapping $\phi : \mathcal{X} \mapsto \mathbb{H}$ associated to some positive definite kernel κ where $\mathbb{H}$ is the corresponding *reproducing kernel Hilbert space* (RKHS), $\kappa(\boldsymbol{x}, \boldsymbol{z}) = \langle \phi(\boldsymbol{x}), \phi(\boldsymbol{z}) \rangle_{\mathbb{H}}$ holds for any $\boldsymbol{x}$ and $\boldsymbol{z}$.

2.1 Optimal Margin Distribution Machine

The traditional large margin based methods maximize the minimum margin, and the obtained decision boundary is only determined by a small number of instances with the minimum margin [Schölkopf and Smola, 2001], which may hurt the generalization performance.

On the other hand, ODM explicitly optimizes the margin distribution. Given a labeled data set $\{(\boldsymbol{x}_i, y_i)\}_{i\in[M]}$, ODM is formalized by maximizing the margin mean and minimizing the margin variance simultaneously:

$$\min_{\boldsymbol{w},\xi_i,\epsilon_i} p(\boldsymbol{w}) = \frac{1}{2}\|\boldsymbol{w}\|^2 + \frac{\lambda}{2M}\sum_{i\in[M]}\frac{\xi_i^2 + \upsilon\epsilon_i^2}{(1-\theta)^2}$$
$$\text{s.t. } 1-\theta-\xi_i \le y_i\boldsymbol{w}^\top\phi(\boldsymbol{x}_i) \le 1+\theta+\epsilon_i,\ \forall i \in [M],$$

where the margin mean has been fixed as 1 since scaling $\boldsymbol{w}$ does not affect the decision boundary, the hyperparameter λ is to balance the regularization and empirical loss, the hyperparameter υ is for trading-off the two different kinds of deviation from margin mean, and the hyperparameter θ is introduced to tolerate small deviations no more than θ.

By introducing the Lagrange multipliers $\boldsymbol{\zeta}, \boldsymbol{\beta} \in \mathbb{R}_+^M$ for the $2M$ inequality constraints respectively, the dual problem of ODM is

$$\min_{\boldsymbol{\zeta},\boldsymbol{\beta}\in\mathbb{R}_+^M} d(\boldsymbol{\zeta},\boldsymbol{\beta}) = \frac{1}{2}(\boldsymbol{\zeta}-\boldsymbol{\beta})^\top\mathbf{Q}(\boldsymbol{\zeta}-\boldsymbol{\beta}) + \frac{Mc}{2}(\upsilon\|\boldsymbol{\zeta}\|^2 + \|\boldsymbol{\beta}\|^2) + (\theta-1)\mathbf{1}_M^\top\boldsymbol{\zeta} + (\theta+1)\mathbf{1}_M^\top\boldsymbol{\beta}, \quad (1)$$

where $[\mathbf{Q}]_{ij} = y_i y_j \kappa(\boldsymbol{x}_i, \boldsymbol{x}_j)$ and $c = (1-\theta)^2/\lambda\upsilon$ is a constant. By denoting $\boldsymbol{\alpha} = [\boldsymbol{\zeta}; \boldsymbol{\beta}]$, the dual ODM can be rewritten as a standard convex QP problem:

$$\min_{\boldsymbol{\alpha}\in\mathbb{R}_+^{2M}} f(\boldsymbol{\alpha}) = \frac{1}{2}\boldsymbol{\alpha}^\top\mathbf{H}\boldsymbol{\alpha} + \boldsymbol{b}^\top\boldsymbol{\alpha}, \quad (2)$$

where

$$\mathbf{H} = \begin{bmatrix} \mathbf{Q} + Mc\upsilon\mathbf{I} & -\mathbf{Q} \\ -\mathbf{Q} & \mathbf{Q} + Mc\mathbf{I} \end{bmatrix}, \quad \boldsymbol{b} = \begin{bmatrix} (\theta-1)\mathbf{1}_M \\ (\theta+1)\mathbf{1}_M \end{bmatrix}.$$

Notice that Eqn. (2) only involves $2M$ decoupled box constraints $\boldsymbol{\alpha} \succeq \mathbf{0}$, thus it can be efficiently solved by a dual coordinate descent method [Zhang and Zhou, 2019]. To be specific, in each iteration, only one variable is selected to update while other variables are kept as constants, which yields the following univariate QP problem of t:

$$\min_t f(\boldsymbol{\alpha} + t\boldsymbol{e}_i) = \frac{1}{2}[\mathbf{H}]_{ii}t^2 + [\nabla f(\boldsymbol{\alpha})]_i t + f(\boldsymbol{\alpha}), \quad (3)$$

with a closed-form solution $\max([\boldsymbol{\alpha}]_i - [\nabla f(\boldsymbol{\alpha})]_i/[\mathbf{H}]_{ii}, 0)$.

3 Proposed Method

SODM works in distributed data level, i.e., dividing the data into partitions on which local models are trained and used to find the larger local or global models. For simplicity, we assume initially there are $K = p^L$ partitions with the same cardinality m, i.e., $m = M/K$. The data set $\{(\boldsymbol{x}_i, y_i)\}_{i\in[M]}$ are ordered so that the first m instances are on the first partition, and the second m instances are on the second partition, etc. That is for any instance $(\boldsymbol{x}_i, y_i)$, the index of partition to which it belongs is $P(i) = \lceil i/m \rceil$ where $\lceil\cdot\rceil$ is the ceil function.

Suppose $\{(\boldsymbol{x}_i^{(k)}, y_i^{(k)})\}_{i\in[m]}$ is the data of the k-th partition, the local ODM trained on it is [cf. Eqn. (1)]

$$\min_{\boldsymbol{\zeta}_k,\boldsymbol{\beta}_k\in\mathbb{R}_+^m} d_k(\boldsymbol{\zeta}_k,\boldsymbol{\beta}_k) = \frac{1}{2}(\boldsymbol{\zeta}_k-\boldsymbol{\beta}_k)^\top\mathbf{Q}^{(k)}(\boldsymbol{\zeta}_k-\boldsymbol{\beta}_k) + \frac{mc}{2}(\upsilon\|\boldsymbol{\zeta}_k\|^2 + \|\boldsymbol{\beta}_k\|^2) + (\theta-1)\mathbf{1}_m^\top\boldsymbol{\zeta}_k + (\theta+1)\mathbf{1}_m^\top\boldsymbol{\beta}_k,$$

where $[\mathbf{Q}^{(k)}]_{ij} = y_i^{(k)} y_j^{(k)} \kappa(\boldsymbol{x}_i^{(k)}, \boldsymbol{x}_j^{(k)})$. This problem can be rewritten as a standard convex QP problem in the same manner as Eqn. (2), and efficiently solved by dual coordinate descent method as Eqn. (3).

Algorithm 1 SODM

Input: Data set $\mathcal{D} = \{(\boldsymbol{x}_i, y_i)\}_{i\in[M]}$, partition control parameter p, number of stratums S, number of iterations L.
Output: The dual solution.

```
Initialize S stratums C_1, ..., C_S by Eqn. (7)-(8).
Initialize partitions D_1, ..., D_{p^L} by sampling without replacement from stratums C_1, ..., C_S.
Initialize α_1, ..., α_{p^L} as 0.
for l = L, ..., 1 do
    if all α_1, ..., α_{p^l} converge then
        return [α_1; ...; α_{p^l}].
    end if
    for k = 1, ..., p^l do
        Solve the local ODM on D_k by dual coordinate descent with α_k as the initial solution.
        if k ≡ 0 (mod p) then
            Form new D_{k/p} by merging D_{k-p+1}, ..., D_k.
            α_{k/p} = [α_{k-p+1}; ...; α_k].
        end if
    end for
end for
return [α_1; ...; α_p].
```

Once the parallel training of p^L local ODMs are completed, we get p solutions. Then we merge every p partitions to form $K/p = p^{L-1}$ larger partitions. On each larger partition, a new local ODM is trained again by dual coordinate descent method, but the optimization procedure is not executed from the scratch. Instead, the previous p solutions are concatenated as the initial point of the optimization. By our proposed novel partition strategy in Section 3.2, this concatenated solution is already a good approximation to the optimal solution thus converges much faster. The above procedure is repeated until the solution converges or all the partitions are merged together. Algorithm 1 summarizes the pseudo-code of SODM.

3.1 Convergence

In this section, we present a theorem to guarantee the convergence of the proposed method. Notice that the optimization variables on each partition are decoupled, they can be jointly optimized by the following problem [cf. Eqn. (1)]

$$\min_{\boldsymbol{\zeta},\boldsymbol{\beta}\in\mathbb{R}^M_+} \widetilde{d}(\boldsymbol{\zeta},\boldsymbol{\beta}) = \frac{1}{2}(\boldsymbol{\zeta}-\boldsymbol{\beta})^\top \widetilde{\mathbf{Q}}(\boldsymbol{\zeta}-\boldsymbol{\beta}) + \frac{mc}{2}(\upsilon\|\boldsymbol{\zeta}\|^2 + \|\boldsymbol{\beta}\|^2) + (\theta-1)\mathbf{1}_M^\top\boldsymbol{\zeta} + (\theta+1)\mathbf{1}_M^\top\boldsymbol{\beta}, \quad (4)$$

where $\widetilde{\mathbf{Q}} = \mathrm{diag}(\mathbf{Q}^{(1)}, \ldots, \mathbf{Q}^{(K)})$ is a block diagonal matrix. It can be seen that the smaller the K, the more close the Eqn. (4) to ODM, and when $K = 1$, it exactly degenerates to ODM. Therefore, SODM deals with ODM by solving a series of problems which approaches to it, and the solution of former problems can be helpful for the optimization of the latter ones.

Theorem 1. *Suppose the optimal solutions of ODM and its approximate problem, i.e., Eqn.* (4)*, are* $\boldsymbol{\alpha}^\star = [\boldsymbol{\zeta}^\star; \boldsymbol{\beta}^\star]$ *and* $\widetilde{\boldsymbol{\alpha}}^\star = [\widetilde{\boldsymbol{\zeta}}^\star; \widetilde{\boldsymbol{\beta}}^\star]$*, respectively, then the gaps between these two optimal solutions satisfy*

$$0 \le d(\widetilde{\boldsymbol{\zeta}}^\star, \widetilde{\boldsymbol{\beta}}^\star) - d(\boldsymbol{\zeta}^\star, \boldsymbol{\beta}^\star) \le U^2(Q + M(M-m)c), \quad (5)$$

$$\|\widetilde{\boldsymbol{\alpha}}^\star - \boldsymbol{\alpha}^\star\|^2 \le \frac{U^2}{Mc\upsilon}(Q + M(M-m)c), \quad (6)$$

where $U = \max(\|\boldsymbol{\alpha}^\star\|_\infty, \|\widetilde{\boldsymbol{\alpha}}^\star\|_\infty)$ *upperbounds the infinity norm of solutions, and* $Q = \sum_{i,j:P(i)\neq P(j)} |[\mathbf{Q}]_{ij}|$ *is the sum of the absolute values of* $\mathbf{Q}$*'s entries which turn to zero in* $\widetilde{\mathbf{Q}}$*.*

Due to the page limitations, we only provide the sketch of proof here. The full proof can be found in arXiv version [1].

Proofsketch. The left-hand side of the Eqn. (5) is due to the optimality of $\boldsymbol{\zeta}^\star$ and $\boldsymbol{\beta}^\star$.

By comparing the definition of $d(\boldsymbol{\zeta},\boldsymbol{\beta})$ in Eqn. (1) and $\widetilde{d}(\boldsymbol{\zeta},\boldsymbol{\beta})$ in Eqn. (4), we can find that the only differences are the change of $\mathbf{Q}$ to $\widetilde{\mathbf{Q}}$ and M to m. Therefore the gap between $d(\boldsymbol{\zeta}^\star,\boldsymbol{\beta}^\star)$ and $\widetilde{d}(\boldsymbol{\zeta}^\star,\boldsymbol{\beta}^\star)$ can be upper bounded by U and Q. The gap between $d(\widetilde{\boldsymbol{\zeta}}^\star,\widetilde{\boldsymbol{\beta}}^\star)$ and $\widetilde{d}(\widetilde{\boldsymbol{\zeta}}^\star,\widetilde{\boldsymbol{\beta}}^\star)$ can be upper bounded in the same manner. Combining these together with $\widetilde{d}(\widetilde{\boldsymbol{\zeta}}^\star,\widetilde{\boldsymbol{\beta}}^\star) \le \widetilde{d}(\boldsymbol{\zeta}^\star,\boldsymbol{\beta}^\star)$ can yield the right-hand side of the Eqn. (5).

Notice that $f(\widetilde{\boldsymbol{\alpha}}^\star)$ is a quadratic function, hence besides the gradient $\boldsymbol{g}$ and Hessian matrix $\mathbf{H}$, all its higher derivatives vanish, and it can be precisely expanded at $\boldsymbol{\alpha}^\star$ as

$$f(\boldsymbol{\alpha}^\star) + \boldsymbol{g}^\top(\widetilde{\boldsymbol{\alpha}}^\star - \boldsymbol{\alpha}^\star) + \frac{1}{2}(\widetilde{\boldsymbol{\alpha}}^\star - \boldsymbol{\alpha}^\star)^\top\mathbf{H}(\widetilde{\boldsymbol{\alpha}}^\star - \boldsymbol{\alpha}^\star),$$

in which $\boldsymbol{g}^\top(\widetilde{\boldsymbol{\alpha}}^\star - \boldsymbol{\alpha}^\star)$ is nonnegative according to the the first order optimality condition. Furthermore, $\mathbf{H}$ can be lower bounded by the sum of a positive semidefinite matrix and a scalar matrix:

$$\mathbf{H} \succeq \begin{bmatrix} \mathbf{Q} & -\mathbf{Q} \\ -\mathbf{Q} & \mathbf{Q} \end{bmatrix} + Mc\upsilon \begin{bmatrix} \mathbf{I} & \\ & \mathbf{I} \end{bmatrix}.$$

By putting all these together, we can show that $\|\widetilde{\boldsymbol{\alpha}}^\star - \boldsymbol{\alpha}^\star\|^2$ is upper bounded by $f(\widetilde{\boldsymbol{\alpha}}^\star) - f(\boldsymbol{\alpha}^\star)$, i.e., $d(\widetilde{\boldsymbol{\zeta}}^\star,\widetilde{\boldsymbol{\beta}}^\star) - d(\boldsymbol{\zeta}^\star,\boldsymbol{\beta}^\star)$, and with the right-hand side of the Eqn. (5), we can derive the Eqn. (6). □

This theorem indicates that the gap between the optimal solutions and the suboptimal solutions obtained in each iteration depends on $M-m$ and Q. As the iteration going on, the partitions become larger and larger, then the number of instances m on each partition approaches to the total number of instances M; on the other hand, the matrix $\widetilde{\mathbf{Q}}$ approaches to $\mathbf{Q}$ which makes Q decrease. Therefore, the solution obtained in each iteration of SODM is getting closer and closer to that of ODM, that is to say, our proposed algorithm converges.

3.2 Partition Strategy

In this section we detail the partition strategy. It can significantly affect the optimization efficiency thus plays a more important role in our proposed method. Up to now, most partition strategies utilize the clustering algorithms to form

[1] https://arxiv.org/abs/2305.04837

the partitions. For example, Hsieh *et al.* [2014] regards each cluster of the kernel k-means as a partition. However, ODM heavily depends on the mean and variance of the training data. Directly treating clusters as partitions will lead to huge difference between the distribution of each partition and the whole data, and consequently huge gap between the local solutions and global solution.

To preserve the original distribution possibly, we borrow the idea from stratified sampling, i.e., we first divide the data set into some homogeneous stratums, and then apply random sampling within each stratum. To be specific, suppose the goal is to generate K partitions. We first choose S landmark points $\{\phi(\boldsymbol{z}_s)\}_{s\in[S]}$ in RKHS, and then construct one stratum for each landmark point by assigning the rest of instances to the stratum in which its nearest landmark point lies, i.e., the index of stratum containing $\boldsymbol{x}_i$ is

$$\varphi(i) = \operatorname*{argmin}_{s\in[S]} \|\phi(\boldsymbol{x}_i) - \phi(\boldsymbol{z}_s)\|. \tag{7}$$

For each stratum $\mathcal{C}_s$, we equally divide it into K pieces by random sampling without replacement and take one piece from each stratum to form a partition, hence totally K partitions are created.

The remaining question is how to select these landmark points. Obviously, they should be representative enough to sketch the whole data distribution. To this end, we exploit the minimal principal angle between different stratum:

$$\tau = \min_{i\neq j} \left\{ \arccos \frac{\langle \phi(\boldsymbol{x}), \phi(\boldsymbol{z})\rangle}{\|\phi(\boldsymbol{x})\|\|\phi(\boldsymbol{z})\|} \;\middle|\; \boldsymbol{x}\in\mathcal{C}_i, \boldsymbol{z}\in\mathcal{C}_j \right\}.$$

Apparently, the larger the angle, the higher variation among the stratums, and the more representative each partition is, which is strictly described by the following theorem.

Theorem 2. *For shift-invariant kernel κ with $\kappa(\boldsymbol{x},\boldsymbol{z}) = \kappa(\boldsymbol{x}-\boldsymbol{z})$, assume $\kappa(0) = r^2$, that is $\|\phi(\boldsymbol{x})\| = r$ for any $\boldsymbol{x}$. With the partition strategy described above, we have*

$$d_k(\boldsymbol{\zeta}_k, \boldsymbol{\beta}_k) - d(\boldsymbol{\zeta}^\star, \boldsymbol{\beta}^\star) \leq U^2M^2c + 2UM + \frac{U^2}{2}(M^2r^2 + r^2\cos\tau(2C - M^2)),\ \forall k\in[K],$$

where $C = \sum_{i,j\in[M]} 1_{\varphi(i)\neq\varphi(j)}$, and U is the same with Theorem 1.

Proofsketch. We construct the auxiliary data set $\widetilde{\mathcal{D}}_k$ by repeating each instance in $\mathcal{D}_k$ for K times, and then show that primal ODM on $\widetilde{\mathcal{D}}_k$ and $\mathcal{D}_k$ have the same optimal objective. Since the strong duality theorem holds for ODM, we have $d_k(\boldsymbol{\zeta}_k, \boldsymbol{\beta}_k) = p_k(\boldsymbol{w}_k) = \widetilde{p}_k(\boldsymbol{w}) = \widetilde{d}_k(\widetilde{\boldsymbol{\zeta}}_k, \widetilde{\boldsymbol{\beta}}_k)$. Next we decompose $\widetilde{d}_k(\widetilde{\boldsymbol{\zeta}}_k, \widetilde{\boldsymbol{\beta}}_k) - d(\boldsymbol{\zeta}^\star, \boldsymbol{\beta}^\star)$ into

$$\frac{1}{2}(\widetilde{\boldsymbol{\zeta}}_k - \widetilde{\boldsymbol{\beta}}_k)^\top \widetilde{\mathbf{Q}}_k(\widetilde{\boldsymbol{\zeta}}_k - \widetilde{\boldsymbol{\beta}}_k) - \frac{1}{2}(\boldsymbol{\zeta}^\star - \boldsymbol{\beta}^\star)^\top \mathbf{Q}(\boldsymbol{\zeta}^\star - \boldsymbol{\beta}^\star),$$

and

$$\frac{Mc\upsilon}{2}(\|\widetilde{\boldsymbol{\zeta}}_k\|^2 - \|\boldsymbol{\zeta}^\star\|^2) + \frac{Mc}{2}(\|\widetilde{\boldsymbol{\beta}}_k\|^2 - \|\boldsymbol{\beta}^\star\|^2) + (\theta-1)\mathbf{1}_M^\top(\widetilde{\boldsymbol{\zeta}}_k - \boldsymbol{\zeta}^\star) + (\theta+1)\mathbf{1}_M^\top(\widetilde{\boldsymbol{\beta}}_k - \boldsymbol{\beta}^\star).$$

Putting the upper bounds of these two terms together can conclude the proof. □

In this theorem, we derive an upper bound of the gap between the optimal objective value on $\mathcal{D}$ and $\mathcal{D}_k$. Notice that $2C > M^2$ holds for any $s\in[S]$ when $|\mathcal{C}_s| < M/2$ is satisfied, a quite mild condition, thus we can get more approximate solution in each partition by maximizing the minimal principal angle τ in RKHS.

Unfortunately, the resultant maximization problem is difficult to solve, so we can hardly acquire the optimal landmark points. But notice that the Gram matrix formed by landmark points should be diagonally dominant and the more strict the better, we can resort to maximizing its determinant. Specifically, suppose $\boldsymbol{z}_1, ..., \boldsymbol{z}_s$ are given, we seek $\boldsymbol{z}_{s+1}$ to maximize

$$\begin{vmatrix} \mathbf{K}_{s,s} & \mathbf{K}_{s,s+1} \\ \mathbf{K}_{s,s+1}^\top & \kappa(\boldsymbol{z}_{s+1}, \boldsymbol{z}_{s+1}) \end{vmatrix} = r^2(r^2 - \mathbf{K}_{s,s+1}^\top \mathbf{K}_{s,s}^{-1}\mathbf{K}_{s,s+1}),$$

where $\mathbf{K}_{s,s}\in\mathbb{R}^{s\times s}$ is the Gram matrix formed by $\boldsymbol{z}_1, ..., \boldsymbol{z}_s$, and $\mathbf{K}_{s,s+1} = [\kappa(\boldsymbol{z}_{s+1}, \boldsymbol{z}_1); ...; \kappa(\boldsymbol{z}_{s+1}, \boldsymbol{z}_s)]$ is a column vector. The equality holds due to the Schur's complement. As for $\boldsymbol{z}_1$, since any choice makes no difference, we can directly set it as $\boldsymbol{x}_1$, and generate other landmark points iteratively via

$$\boldsymbol{z}_{s+1} = \operatorname*{argmin}_{\boldsymbol{z}_{s+1}} \mathbf{K}_{s,s+1}^\top \mathbf{K}_{s,s}^{-1}\mathbf{K}_{s,s+1},\ \forall s\in[S-1]. \tag{8}$$

It is noteworthy that each partition generated by our proposed strategy extracts proportional instances from each stratum, thus preserves the distribution. Besides, compared with other partition strategies based on k-means [Singh *et al.*, 2017], we consider both the original feature space and the situation when data can hardly be linearly separated. Last but not least, our partition strategy is computationally efficient.

3.3 Acceleration for Linear Kernel

Dual coordinate descent method requires too many computation and storage resources, mainly caused by the enormous kernel matrix. But fortunately, when linear kernel is applied, we can directly solve the primal form of ODM, avoiding the computation and storage of kernel matrix.

The objective function of ODM is differentiable and the gradient of $p(\boldsymbol{w})$ on instance $(\boldsymbol{x}_i, y_i)$ is

$$\nabla p_i(\boldsymbol{w}) = \boldsymbol{w} + \frac{\lambda(y_i\boldsymbol{w}^\top\boldsymbol{x}_i + \theta - 1)y_i\boldsymbol{x}_i 1_{i\in\mathcal{I}_1}}{(1-\theta)^2} + \frac{\lambda\upsilon(y_i\boldsymbol{w}^\top\boldsymbol{x}_i - \theta - 1)y_i\boldsymbol{x}_i 1_{i\in\mathcal{I}_2}}{(1-\theta)^2},$$

where $\mathcal{I}_1 = \{i \mid y_i\boldsymbol{w}^\top\boldsymbol{x}_i < 1-\theta\}$ and $\mathcal{I}_2 = \{i \mid y_i\boldsymbol{w}^\top\boldsymbol{x}_i > 1+\theta\}$. *Distributed SVRG* (DSVRG) [Lee *et al.*, 2017] can be exploited in this scenario. It generates a series of extra auxiliary data sets sampling from the original data set without replacement which share the same distribution as the whole data set, so that an unbiased estimation of the gradient can be acquired. In each iteration, all nodes (partitions) are joined together to compute the full gradient first. Then each node performs the iterative update of SVRG in serial in a "round robin" fashion, i.e., let all nodes stay idle except one node performing a certain steps of iterative updates using its local auxiliary data and passing the solution to the next node. Algorithm 2 summarizes the process of DSVRG for SODM.

Data sets	gisette	svmguide1	phishing	a7a	cod-rna	ijcnn1	skin-nonskin	SUSY
#Instance	7,000	7,089	11,055	32,561	59,535	141,691	245,057	5,000,000
#Feature	5,000	4	68	123	8	22	3	18

Table 1: Data set statistics

Data sets	ODM	Ca-ODM		DiP-ODM		DC-ODM		SODM	
	Acc.	Acc.	Time	Acc.	Time	Acc.	Time	Acc.	Time
gisette	.976	.957	90.22	.970	68.02	.964	70.44	**.972**	59.89
svmguide1	.970	.872	38.90	.903	35.25	.943	50.11	**.944**	28.74
phishing	.941	.880	49.60	.901	52.61	.936	59.47	**.938**	25.22
a7a	.882	.824	68.36	.813	61.24	.815	106.51	**.838**	32.67
cod-rna	N/A	.892	499.38	.905	532.68	.931	400.61	**.933**	55.41
ijcnn1	N/A	.889	185.20	.893	182.71	.915	226.26	**.927**	40.32
skin-nonskin	N/A	.806	338.73	.830	437.20	**.962**	407.46	.956	283.36
SUSY	N/A	.733	4280.23	.744	5678.66	.747	7009.36	**.760**	1004.33

Table 2: The test accuracy and time cost (in seconds) of different methods using RBF kernel. The best accuracy on each data set is bolded. N/A means the corresponding method does not return results in 48 hours.

Algorithm 2 Accelerated SODM for linear kernel

Input: Data set $\mathcal{D} = \{(\boldsymbol{x}_i, y_i)\}_{i\in[M]}$, number of partitions K, number of stratums S, number of epoch E, step size η.
Output: Solution $\boldsymbol{w}^{(E)}$ at epoch E.

Initialize S stratums $\mathcal{C}_1, \ldots, \mathcal{C}_S$ by Eqn. (7)-(8).
Initialize partitions $\mathcal{D}_1, \ldots, \mathcal{D}_K$ by sampling without replacement from stratums $\mathcal{C}_1, \ldots, \mathcal{C}_S$.
Generate the auxiliary array $\mathcal{R}_1, \ldots, \mathcal{R}_K$ where $\mathcal{R}_j = \{i \mid (\boldsymbol{x}_i, y_i) \in \mathcal{D}_j\}$.
for $l = 0, 1, \ldots, E-1$ **do**
  The center node sends $\boldsymbol{w}^{(l)}$ to each node.
  for each node $j = 1, 2, \ldots, K$ in parallel **do**
    $\boldsymbol{h}_j^{(l)} = \sum_{i\in\mathcal{D}_j} \nabla p_i(\boldsymbol{w}^{(l)})$.
  end for
  The center node computes $\boldsymbol{h}^{(l)} = \frac{1}{M}\sum_{j=1}^{K} \boldsymbol{h}_j^{(l)}$ and sends it to each node.
  $\boldsymbol{w}_0^{(l+1)} = \boldsymbol{w}^{(l)}$.
  $t = 0$.
  for $j = 1, 2, \ldots, K$ **do**
    Sample instances $(\boldsymbol{x}_i, y_i)$ from $\mathcal{D}_j$ where $i \in \mathcal{R}_j$.
    $\boldsymbol{w}_{t+1}^{(l+1)} = \boldsymbol{w}_t^{(l+1)} - \eta(\nabla p_i(\boldsymbol{w}_t^{(l+1)}) - \nabla p_i(\boldsymbol{w}^{(l)}) + \boldsymbol{h}^{(l)})$.
    $t = t + 1$.
    $\mathcal{R}_j = \mathcal{R}_j \backslash i$.
    if $\mathcal{R}_j = \emptyset$ **then**
      Continue.
    end if
  end for
  $\boldsymbol{w}^{(l+1)} = \boldsymbol{w}_t^{(l+1)}$.
end for
return $\boldsymbol{w}^{(E)}$.

4 Experiments

In this section, we evaluate the proposed algorithms by comparing with other SOTA scalable QP solvers.

4.1 Setup

All the experiments are performed on eight real-world data sets. The statistics of these data sets are summarized in Table 1. All features are normalized into the interval $[0, 1]$. For each data set, eighty percent of instances are randomly selected as training data, while the rest are testing data. All the experiments are performed on a Spark [Zaharia *et al.*, 2012] cluster with one master and five workers. Each machine is equipped with 16 Intel Xeon E5-2670 CPU cores and 64GB RAM. Our implementation are available on Github [2].

SODM is compared with three SOTA scalable QP solvers, i.e., Cascade approach (Ca-ODM) [Graf *et al.*, 2004], DiP approach (DiP-ODM) [Singh *et al.*, 2017], and DC approach (DC-ODM) [Hsieh *et al.*, 2014]. Besides, to evaluate the efficiency of the accelerated SODM for linear kernel, two SOTA gradient based methods are implementd, i.e., SVRG method (ODM$_{svrg}$) [Johnson and Zhang, 2013] and CSVRG method (ODM$_{csvrg}$) [Tan *et al.*, 2019].

4.2 Results with RBF Kernel

Figure 1 presents the test accuracy and time cost of different methods with RBF kernel. It can be seen that SODM performs significantly better than other methods. Specifically, SODM achieves the best test accuracy on 7 data sets and just slightly worse than DC-ODM on data set skin-nonskin. As for time cost, SODM achieves the fastest training speed on all data sets. The detailed test accuracy and time cost are presented in Table 2. The time cost and test accuracy with corresponding SVM can be found in arXiv version.

[2]https://github.com/CGCL-codes/SODM

Data sets	ODM	Ca-ODM		DiP-ODM		DC-ODM		SODM	
	Acc.	Acc.	Time	Acc.	Time	Acc.	Time	Acc.	Time
gisette	.972	.953	82.35	.966	74.36	**.968**	66.32	**.968**	28.57
svmguide1	.964	.863	35.27	.898	40.52	**.933**	41.85	.931	18.93
phishing	.937	.894	33.84	.921	38.60	.926	29.04	**.933**	11.75
a7a	.850	.795	47.59	.831	59.17	.833	85.42	**.847**	16.41
cod-rna	.938	.882	435.19	.894	434.77	.890	331.46	**.934**	17.29
ijcnn1	.913	.896	228.43	.903	208.81	.883	214.66	**.920**	21.15
skin-nonskin	.917	.796	158.12	.903	256.78	**.922**	340.30	.909	21.15
SUSY	.774	.734	3790.37	.738	3829.23	.747	7095.32	**.760**	178.92

Table 3: The test accuracy and time cost (in seconds) of different methods using linear kernel. The best accuracy on each data set is bolded.

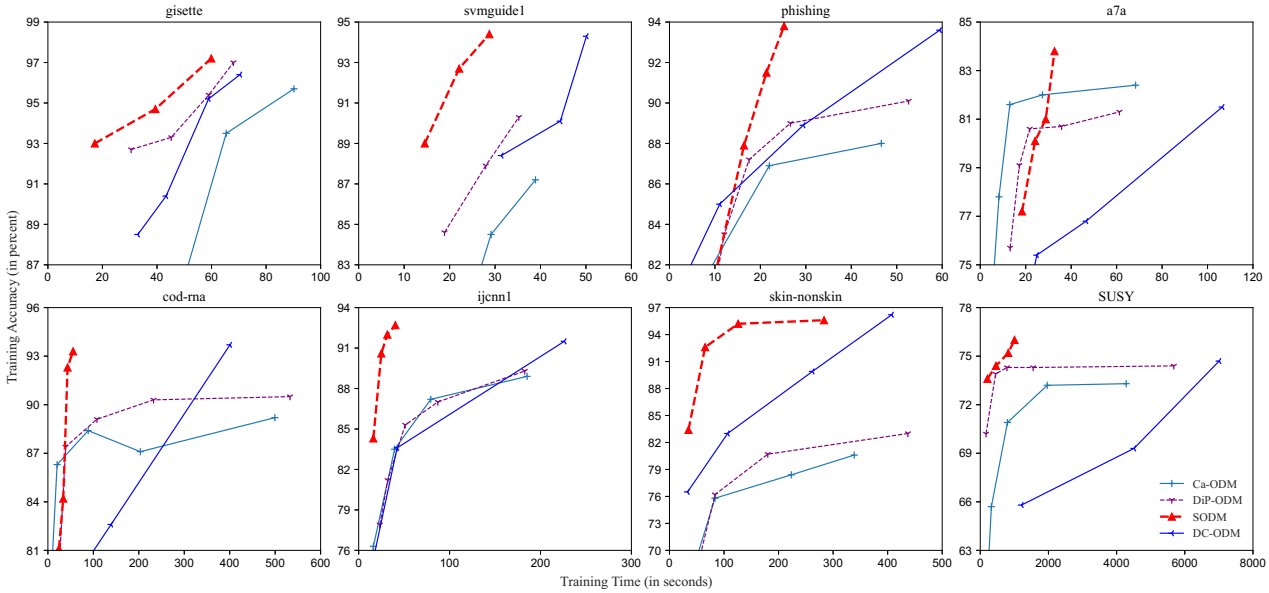

Figure 1: Comparisons of different methods using RBF kernel. Each point indicates the result when stop at different levels.

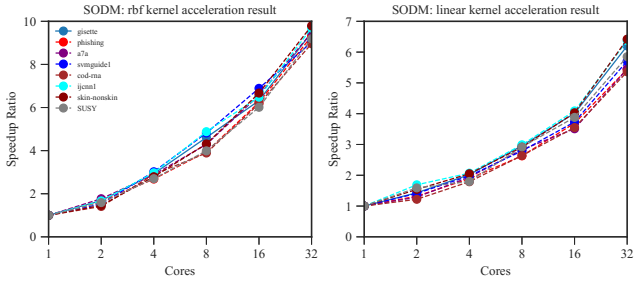

Figure 2: Training speedup ratio with cores increasing from 1 to 32 for SODM

4.3 Results with Linear Kernel

Figure 3 presents the test accuracy and time cost of different methods with linear kernel. It can be seen that SODM shows highly competitive performance compared with other methods. Specifically, SODM achieves the best test accuracy on 6 data sets and just slightly worse than DC-ODM on data set svmguide1 and skin-nonskin. As for time cost, SODM achieves faster training speed on all data sets. The detailed test accuracy and time cost are presented in Table 3. In Figure 2, we show the training speedup ratio with cores increasing from 1 to 32 for linear kernel and RBF kernel, respectively. When 32 cores used, RBF kernel SODM achieves more than 9 times training speedup while linear kernel SODM achieves over 5 times training speedup.

4.4 Comparison with Gradient Based Methods

Figure 4 compares the test accuracy and time cost between our acceleration method and other gradient based methods. We observe that our method can get competitive result. Meanwhile, our method achieves over 5 times faster speed than other methods. This indicates that our scalable acceleration method can achieve great training speed while hold the generalization performance.

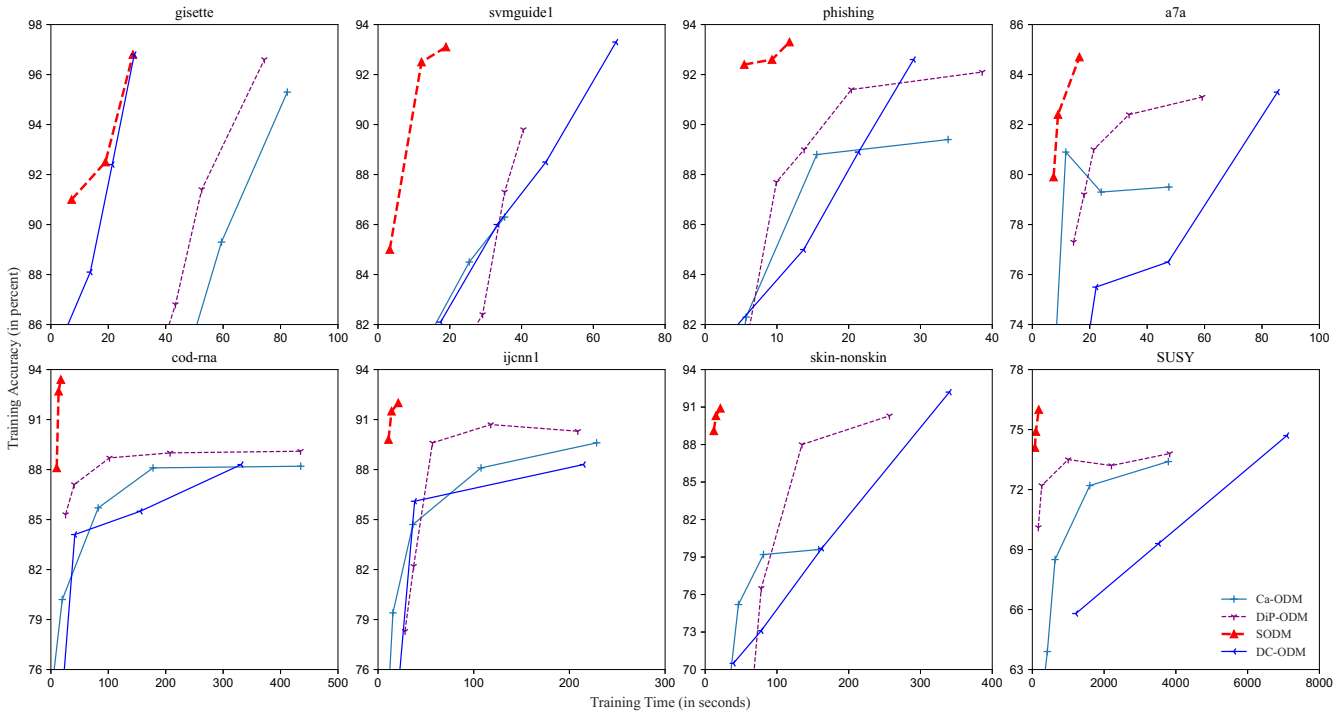

Figure 3: Comparisons of different methods using linear kernel. Each point of SODM indicates the result when every one third of epochs executed. Other points indicate the result stop at different levels.

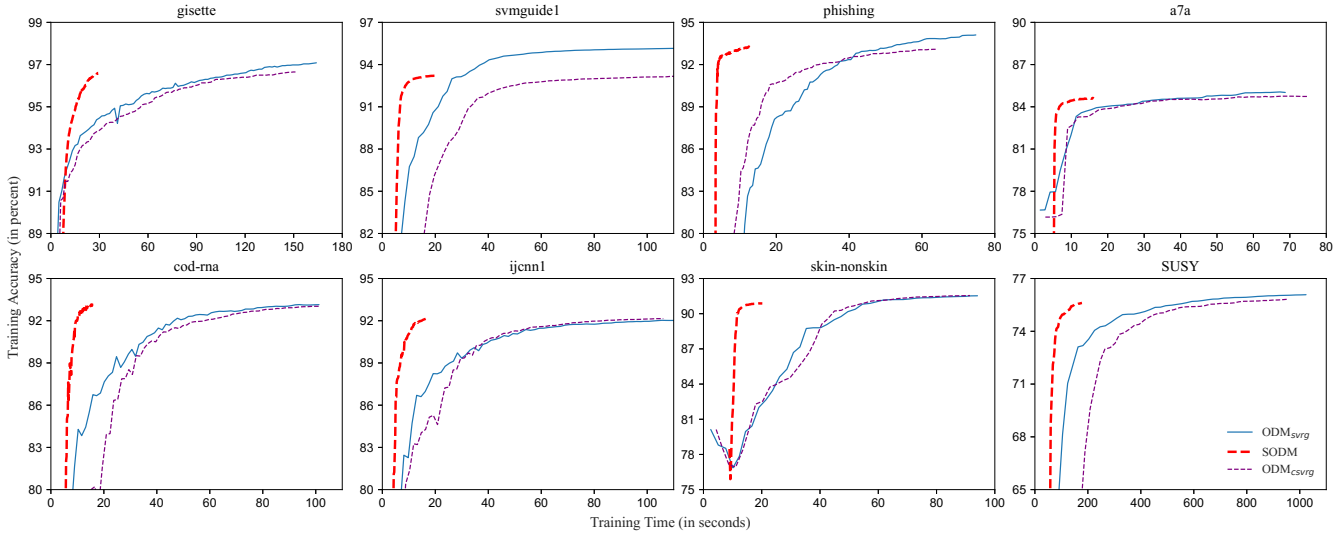

Figure 4: Comparisons of different gradient based methods

5 Conclusion

Although lots of works have been proposed to solve QP problems, these off-the-shelf solvers usually ignore the intrinsic structure of the optimization problem, thus can hardly achieve the greatest efficiency when directly applied to ODM. We propose a scalable ODM with a novel partition strategy, which can retain the first- and second- order statistics in both the original instance space and the RKHS, leading to significant speedup of training. In addition, an accelerating method is implemented to further improve the training when linear kernel is used. As shown in the experiments, SODM has great superiority to other scalable QP solvers in terms of both generalization performance and time cost. In the future, we will consider the circumstance in which data is located on different devices and can not be gathered together due to the limited bandwidth or user privacy.

Acknowledgments

This work was supported in part by the National Key R&D Program of China under Grant 2020AAA0108501, the National Natural Science Foundation of China under Grant 62006088, and the Key R&D Program of Hubei under Grant 2020BAA020.

References

[Boyd *et al.*, 2010] Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato, and Jonathan Eckstein. Distributed Optimization and Statistical Learning via the Alternating Direction Method of Multipliers. *Foundations and Trends in Machine Learning*, 3(1):1–122, 2010.

[Cao *et al.*, 2006] Lijuan Cao, Selvaraj Sathiya Keerthi, Chong Jin Ong, Jianqiu Zhang, Uvaraj Periyathamby, Xiuju Fu, and Henry P. Lee. Parallel sequential minimal optimization for the training of support vector machines. *IEEE Transactions on Neural Networks*, 17(4):1039–1049, 2006.

[Cao *et al.*, 2021] Nan Cao, Teng Zhang, and Hai Jin. Partial Multi-Label Optimal Margin Distribution Machine. In *Proceedings of the 30th International Joint Conference on Artificial Intelligence*, pages 2198–2204, Montreal-themed virtual reality, 2021.

[Cao *et al.*, 2022] Nan Cao, Teng Zhang, Xuanhua Shi, and Hai Jin. Posistive-Unlabeled Learning via Optimal Transport and Margin Distribution. In *Proceedings of the 31st International Joint Conference on Artificial Intelligence*, pages 2836–2842, Vienna, Austria, 2022.

[Cheng *et al.*, 2017] Fanyong Cheng, Jing Zhang, Cuihong Wen, Zhaohua Liu, and Zuoyong Li. Large cost-sensitive margin distribution machine for imbalanced data classification. *Neurocomputing*, 224:45–57, 2017.

[Forero *et al.*, 2010] Pedro A. Forero, Alfonso Cano, and Georgios B. Giannakis. Consensus-Based Distributed Support Vector Machines. *Journal of Machine Learning Research*, 11:1663–1701, 2010.

[Gao and Zhou, 2013] Wei Gao and Zhi-Hua Zhou. On the doubt about margin explanation of boosting. *Artificial Intelligence*, 203:1–18, 2013.

[Graf *et al.*, 2004] Hans Peter Graf, Eric Cosatto, Leon Bottou, Igor Dourdanovic, and Vladimir Vapnik. Parallel Support Vector Machines: The Cascade SVM. In *Advances in Neural Information Processing Systems*, pages 521–528, Vancouver, Canada, 2004.

[Grønlund *et al.*, 2019] Allan Grønlund, Lior Kamma, Kasper Green Larsen, Alexander Mathiasen, and Jelani Nelson. Margin-Based Generalization Lower Bounds for Boosted Classifiers. In *Advances in Neural Information Processing Systems*, pages 11963–11972, Vancouver, Canada, 2019.

[Hsieh *et al.*, 2014] Cho-Jui Hsieh, Si Si, and Inderjit Singh Dhillon. A Divide-and-Conquer Solver for Kernel Support Vector Machines. In *Proceedings of the 31st International Conference on Machine Learning*, pages 566–574, Beijing, China, 2014.

[Johnson and Zhang, 2013] Rie Johnson and Tong Zhang. Accelerating Stochastic Gradient Descent using Predictive Variance Reduction. In *Advances in Neural Information Processing Systems*, pages 315–323, Lake Tahoe, NV, 2013.

[Lee *et al.*, 2017] Jason D. Lee, Qihang Lin, Tengyu Ma, and Tianbao Yang. Distributed Stochastic Variance Reduced Gradient Methods by Sampling Extra Data with Replacement. *Journal of Machine Learning Research*, 18(122):1–43, 2017.

[Loosli *et al.*, 2007] Gaëlle Loosli, Stéphane Canu, and Léon Bottou. Training invariant support vector machines using selective sampling. In Léon Bottou, Olivier Chapelle, Dennis DeCoste, and Jason Weston, editors, *Large-Scale Kernel Machines*, pages 301–320. MIT Press, Cambridge, MA, 2007.

[Luan *et al.*, 2020] Tianxiang Luan, Tingjin Luo, Wenzhang Zhuge, and Chenping Hou. Optimal Representative Distribution Margin Machine for Multi-Instance Learning. *IEEE Access*, 8:74864–74874, 2020.

[Navia-Vazquez *et al.*, 2006] Angel Navia-Vazquez, D. Gutierrez-Gonzalez, Emilio Parrado-Hernandez, and J. J. Navarro-Abellan. Distributed Support Vector Machines. *IEEE Transactions on Neural Networks*, 17(4):1091–1097, 2006.

[Rahimi and Recht, 2007] Ali Rahimi and Benjamin Recht. Random Features for Large-Scale Kernel Machines. In *Advances in Neural Information Processing Systems*, pages 1177–1184, Vancouver, Canada, 2007.

[Rastogi *et al.*, 2020] Reshma Rastogi, Pritam Anand, and Suresh Chandra. Large-margin Distribution Machine-based regression. *Neural Computing and Applications*, 32:3633–3648, 2020.

[Schölkopf and Smola, 2001] Bernhard Schölkopf and Alexander Johannes Smola. *Learning with kernels: support vector machines, regularization, optimization, and beyond*. MIT Press, Cambridge, MA, 2001.

[Si *et al.*, 2017] Si Si, Cho-Jui Hsieh, and Inderjit Singh Dhillon. Memory Efficient Kernel Approximation. *Journal of Machine Learning Research*, 18:1–32, 2017.

[Singh *et al.*, 2017] Dinesh Singh, Debaditya Roy, and Chalavadi Krishna Mohan. DiP-SVM: Distribution Preserving Kernel Support Vector Machine for Big Data. *IEEE Transactions on Big Data*, 3(1):79–90, 2017.

[Tan *et al.*, 2019] Zhi-Hao Tan, Teng Zhang, and Wei Wang. Coreset Stochastic Variance-Reduced Gradient with Application to Optimal Margin Distribution Machine. In *Proceedings of the 33rd AAAI Conference on Artificial Intelligence*, pages 5083–5090, Honolulu, HI, 2019.

[Tan *et al.*, 2020] Zhi-Hao Tan, Peng Tan, Yuan Jiang, and Zhi-Hua Zhou. Multi-label Optimal Margin Distribution Machine. *Machine Learning*, 109(3):623–642, 2020.

[Williams and Seeger, 2001] Christopher Williams and Matthias Seeger. Using the Nyström Method to Speed Up Kernel Machines. In *Advances in Neural Information*

Processing Systems, pages 682–688, Cambridge, MA, 2001.

[Yu *et al.*, 2005] Hwanjo Yu, Jiong Yang, Jiawei Han, and Xiaolei Li. Making SVMs scalable to large data sets using hierarchical cluster indexing. *Data Mining and Knowledge Discovery*, 11(3):295–321, 2005.

[Zaharia *et al.*, 2012] Matei Zaharia, Mosharaf Chowdhury, Tathagata Das, Ankur Dave, Justin Ma, Murphy McCauly, Michael Jay Franklin, Scott Shenker, and Ion Stoica. Resilient Distributed Datasets: A Fault-Tolerant Abstraction for In-Memory Cluster Computing. In *Proceedings of the 9th USENIX Symposium on Networked Systems Design and Implementation*, pages 15–28, San Jose, CA, 2012.

[Zhang and Jin, 2020] Teng Zhang and Hai Jin. Optimal Margin Distribution Machine for Multi-Instance Learning. In *Proceedings of the 29th International Joint Conference on Artificial Intelligence*, pages 2383–2389, 2020.

[Zhang and Zhou, 2018a] Teng Zhang and Zhi-Hua Zhou. Optimal margin distribution clustering. In *Proceedings of the 32nd AAAI Conference on Artificial Intelligence*, pages 4474–4481, New Orleans, LA, 2018.

[Zhang and Zhou, 2018b] Teng Zhang and Zhi-Hua Zhou. Semi-supervised optimal margin distribution machines. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, pages 3104–3110, Stockholm, Sweden, 2018.

[Zhang and Zhou, 2019] Teng Zhang and Zhi-Hua Zhou. Optimal Margin Distribution Machine. *IEEE Transactions on Knowledge and Data Engineering*, 32(6):1143–1156, 2019.

[Zhang *et al.*, 2020] Teng Zhang, Peng Zhao, and Hai Jin. Optimal Margin Distribution Learning in Dynamic Environments. In *Proceedings of the 34th AAAI Conference on Artificial Intelligence*, pages 6821–6828, New York, NY, 2020.

[Zhou and Zhou, 2016] Yu-Hang Zhou and Zhi-Hua Zhou. Large Margin Distribution Learning with Cost Interval and Unlabeled data. *IEEE Transactions on Knowledge and Data Engineering*, 28(7):1749–1763, 2016.