Large Margin Distribution Machine

Teng Zhang and Zhi-Hua Zhou
National Key Laboratory for Novel Software Technology
Nanjing University, Nanjing 210023, China
{zhangt, zhouzh}@lamda.nju.edu.cn

ABSTRACT

Support vector machine (SVM) has been one of the most popular learning algorithms, with the central idea of maximizing the *minimum margin*, i.e., the smallest distance from the instances to the classification boundary. Recent theoretical results, however, disclosed that maximizing the minimum margin does not necessarily lead to better generalization performances, and instead, the margin distribution has been proven to be more crucial. In this paper, we propose the Large margin Distribution Machine (LDM), which tries to achieve a better generalization performance by optimizing the margin distribution. We characterize the margin distribution by the first- and second-order statistics, i.e., the margin mean and variance. The LDM is a general learning approach which can be used in any place where SVM can be applied, and its superiority is verified both theoretically and empirically in this paper.

Categories and Subject Descriptors

I.2.6 [**Artificial Intelligence**]: Learning; I.5.2 [**Pattern Recognition**]: Design Methodology—*classifier design and evaluation*; H.2.8 [**Database Management**]: Database Applications—*Data mining*

Keywords

Margin distribution; minimum margin; classification

1. INTRODUCTION

Support Vector Machine (SVM) [5, 26] has always been one of the most successful learning algorithms. The basic idea is to identify a classification boundary having a large margin for all the training examples, 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 [16, 6, 25, 14, 8] on improving SVMs.

KDD'14, August 24–27, 2014, New York, NY, USA.

ACM 978-1-4503-2956-9/14/08...$15.00.
http://dx.doi.org/10.1145/2623330.2623710.

It is well known that SVM can be viewed as a learning approach trying to maximize over training examples the *minimum margin*, i.e., the smallest distance from the examples to the classification boundary, and the margin theory [26] provided a good support to the generalization performance of SVM. It is noteworthy that the margin theory not only plays an important role for SVMs, but also has been extended to interpret the good generalization of many other learning approaches, such as AdaBoost [10], a major representative of ensemble methods [31]. Specifically, Schapire et al. [21] first suggested margin theory to explain the phenomenon that AdaBoost seems resistant to overfitting; soon after, Breiman [4] indicated that the minimum margin is crucial and developed a boosting-style algorithm, Arc-gv, which is able to maximize the minimum margin but with a poor generalization performance. Later, Reyzin et al. [20] found that although Arc-gv tends to produce larger minimum margin, it suffers from a poor margin distribution; they conjectured that the margin distribution, rather than the minimum margin, is more crucial to the generalization performance. Such a conjecture has been theoretically studied [27, 11], and it was recently proven by Gao and Zhou [11]. Moreover, it was disclosed that rather than simply considering a single-point margin, both the margin mean and variance are important [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 this paper, we propose the Large margin Distribution Machine (LDM), which tries to achieve strong generalization performance by optimizing the margin distribution. Inspired by the recent theoretical result [11], we characterize the margin distribution by the first- and second-order statistics, and try to maximize the margin mean and minimize the margin variance simultaneously. For optimization, we propose a dual coordinate descent method for kernel LDM, and propose an averaged stochastic gradient descent (ASGD) method for large scale linear kernel LDM. Comprehensive experiments on twenty regular scale data sets and twelve large scale data sets show the superiority of LDM to SVM and many state-of-the-art methods, verifying that the margin distribution is more crucial for SVM-style learning approaches than minimum margin.

The rest of this paper is organized as follows. Section 2 introduces some preliminaries. Section 3 presents the LDM. Section 4 reports on our experiments. Section 5 discusses about some related works. Finally, Section 6 concludes.

2. PRELIMINARIES

We denote by $\mathcal{X} \in \mathbb{R}^d$ the instance space and $\mathcal{Y} = \{+1, -1\}$ the label set. Let $\mathcal{D}$ be an unknown (underlying) distribution over $\mathcal{X} \times \mathcal{Y}$. A training set of size m

$$S = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \ldots, (\boldsymbol{x}_m, y_m)\},$$

is drawn identically and independently (i.i.d.) according to the distribution $\mathcal{D}$. Our goal is to learn a function which is used to predict the labels for future unseen instances.

For SVMs, f is regarded as a linear model, i.e., $f(\boldsymbol{x}) = \boldsymbol{w}^\top \phi(\boldsymbol{x})$ where $\boldsymbol{w}$ is a linear predictor, $\phi(\boldsymbol{x})$ is a feature mapping of $\boldsymbol{x}$ induced by a kernel k, i.e., $k(\boldsymbol{x}_i, \boldsymbol{x}_j) = \phi(\boldsymbol{x}_i)^\top \phi(\boldsymbol{x}_j)$. According to [5, 26], the margin of instance $(\boldsymbol{x}_i, y_i)$ is formulated as

$$\gamma_i = y_i \boldsymbol{w}^\top \phi(\boldsymbol{x}_i), \forall i = 1, \ldots, m. \tag{1}$$

From [7], it is shown that in separable cases where the training examples can be separated with the zero error, SVM with hard-margin (or Hard-margin SVM),

$$\begin{aligned} \min_{\boldsymbol{w}} \ & \frac{1}{2}\boldsymbol{w}^\top \boldsymbol{w} \\ \text{s.t.} \ & y_i \boldsymbol{w}^\top \phi(\boldsymbol{x}_i) \geq 1, \ i = 1, \ldots, m, \end{aligned}$$

is regarded as the maximization of the minimum margin $\{\min\{\gamma_i\}_{i=1}^m\}$.

In non-separable cases where the training examples cannot be separated with the zero error, SVM with soft-margin (or Soft-margin SVM) is posed,

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

where $\boldsymbol{\xi} = [\xi_1, \ldots, \xi_m]^\top$ measure the losses of instances, and C is a trading-off parameter. There exists a constant $\bar{C}$ such that (2) can be equivalently reformulated as,

$$\begin{aligned} \max_{\boldsymbol{w}} \ & \gamma_0 - \bar{C} \textstyle\sum_{i=1}^m \xi_i \\ \text{s.t.} \ & \gamma_i \geq \gamma_0 - \xi_i, \\ & \xi_i \geq 0, \ i = 1, \ldots, m, \end{aligned}$$

where γ_0 is a relaxed minimum margin, and $\bar{C}$ is the trading-off parameter. Note that γ_0 indeed characterizes the top-p minimum margin [11]; hence, SVMs (with both hard-margin and soft-margin) consider only a single-point margin and have not exploited the whole margin distribution.

3. LDM

In this section, we first formulate the margin distribution, and then present the optimization algorithms and the theoretical guarantee.

3.1 Formulation

The two most straightforward statistics for characterizing the margin distribution are the first- and second-order statistics, that is, the mean and the variance of the margin. Formally, denote $\boldsymbol{X}$ as the matrix whose i-th column is $\phi(\boldsymbol{x}_i)$, i.e., $\boldsymbol{X} = [\phi(\boldsymbol{x}_1) \ldots \phi(\boldsymbol{x}_m)]$, $\boldsymbol{y} = [y_1, \ldots, y_m]^\top$ is a column vector, and $\boldsymbol{Y}$ is a $m \times m$ diagonal matrix with $y_1, \ldots, y_m$ as the diagonal elements. According to the definition in (1), the margin mean is

$$\bar{\gamma} = \frac{1}{m}\sum_{i=1}^m y_i \boldsymbol{w}^\top \phi(\boldsymbol{x}_i) = \frac{1}{m}(\boldsymbol{X}\boldsymbol{y})^\top \boldsymbol{w}, \tag{3}$$

and the margin variance is

$$\begin{aligned} \hat{\gamma} &= \frac{1}{m^2}\sum_{i=1}^m \sum_{j=1}^m (y_i \boldsymbol{w}^\top \phi(\boldsymbol{x}_i) - y_j \boldsymbol{w}^\top \phi(\boldsymbol{x}_j))^2 \\ &= \frac{2}{m^2}(m\boldsymbol{w}^\top \boldsymbol{X}\boldsymbol{X}^\top \boldsymbol{w} - \boldsymbol{w}^\top \boldsymbol{X}\boldsymbol{y}\boldsymbol{y}^\top \boldsymbol{X}^\top \boldsymbol{w}). \end{aligned} \tag{4}$$

Inspired by the recent theoretical result [11], LDM attempts to maximize the margin mean and minimize the margin variance simultaneously.

We first consider a simpler scenario, i.e., the separable cases where the training examples can be separated with the zero error. In these cases, the maximization of the margin mean and the minimization of the margin variance leads to the following hard-margin LDM,

$$\begin{aligned} \min_{\boldsymbol{w}} \ & \frac{1}{2}\boldsymbol{w}^\top \boldsymbol{w} + \lambda_1 \hat{\gamma} - \lambda_2 \bar{\gamma} \\ \text{s.t.} \ & y_i \boldsymbol{w}^\top \phi(\boldsymbol{x}_i) \geq 1, \ i = 1, \ldots, m, \end{aligned}$$

where λ_1 and λ_2 are the parameters for trading-off the margin variance, the margin mean and the model complexity. It's evident that the hard-margin LDM subsumes the hard-margin SVM when λ_1 and λ_2 equal 0.

For the non-separable cases, similar to soft-margin SVM, the soft-margin LDM leads to

$$\begin{aligned} \min_{\boldsymbol{w}, \boldsymbol{\xi}} \ & \frac{1}{2}\boldsymbol{w}^\top \boldsymbol{w} + \lambda_1 \hat{\gamma} - \lambda_2 \bar{\gamma} + C \sum_{i=1}^m \xi_i \\ \text{s.t.} \ & y_i \boldsymbol{w}^\top \phi(\boldsymbol{x}_i) \geq 1 - \xi_i, \\ & \xi_i \geq 0, \ i = 1, \ldots, m. \end{aligned} \tag{5}$$

Similarly, soft-margin LDM subsumes the soft-margin SVM if λ_1 and λ_2 both equal 0. Because the soft-margin SVM often performs much better than the hard-margin one, in the following we will focus on soft-margin LDM and if without clarification, LDM is referred to the soft-margin LDM.

3.2 Optimization

We in this section first present a dual coordinate descent method for kernel LDM, and then present an average stochastic gradient descent (ASGD) method for large scale linear kernel LDM.

3.2.1 Kernel LDM

By substituting (3)-(4), (5) leads to the following quadratic programming problem,

$$\begin{aligned} \min_{\boldsymbol{w}, \boldsymbol{\xi}} \ & \frac{1}{2}\boldsymbol{w}^\top \boldsymbol{w} + \frac{2\lambda_1}{m^2}(m\boldsymbol{w}^\top \boldsymbol{X}\boldsymbol{X}^\top \boldsymbol{w} - \boldsymbol{w}^\top \boldsymbol{X}\boldsymbol{y}\boldsymbol{y}^\top \boldsymbol{X}^\top \boldsymbol{w}) \\ & - \lambda_2 \frac{1}{m}(\boldsymbol{X}\boldsymbol{y})^\top \boldsymbol{w} + C \sum_{i=1}^m \xi_i \\ \text{s.t.} \ & y_i \boldsymbol{w}^\top \phi(\boldsymbol{x}_i) \geq 1 - \xi_i, \\ & \xi_i \geq 0, \ i = 1, \ldots, m. \end{aligned} \tag{6}$$

(6) is often intractable due to the high or infinite dimensionality of $\phi(\cdot)$. Fortunately, inspired by the representer

theorem in [22], the following theorem states that the optimal solution for (6) can be spanned by $\{\phi(\boldsymbol{x}_i), 1 \le i \le m\}$.

THEOREM 1. *The optimal solution $\boldsymbol{w}^*$ for problem (6) admits a representation of the form*

$$\boldsymbol{w}^* = \sum_{i=1}^{m} \alpha_i \phi(\boldsymbol{x}_i) = \boldsymbol{X}\boldsymbol{\alpha}, \tag{7}$$

where $\boldsymbol{\alpha} = [\alpha_1, \ldots, \alpha_m]^\top$ are the coefficients.

PROOF. $\boldsymbol{w}^*$ can be decomposed into a part that lives in the span of $\phi(\boldsymbol{x}_i)$ and an orthogonal part, i.e.,

$$\boldsymbol{w} = \sum_{i=1}^{m} \alpha_i \phi(\boldsymbol{x}_i) + \boldsymbol{v} = \boldsymbol{X}\boldsymbol{\alpha} + \boldsymbol{v}$$

for some $\boldsymbol{\alpha} = [\alpha_1, \ldots, \alpha_m]^\top$ and $\boldsymbol{v}$ satisfying $\phi(\boldsymbol{x}_j)^\top \boldsymbol{v} = 0$ for all j, i.e., $\boldsymbol{X}^\top \boldsymbol{v} = \boldsymbol{0}$. Note that

$$\boldsymbol{X}^\top \boldsymbol{w} = \boldsymbol{X}^\top (\boldsymbol{X}\boldsymbol{\alpha} + \boldsymbol{v}) = \boldsymbol{X}^\top \boldsymbol{X}\boldsymbol{\alpha},$$

so the second and the third terms of (6) are independent of $\boldsymbol{v}$; further note that the constraint is also independent of $\boldsymbol{v}$, thus the last terms of (6) is also independent of $\boldsymbol{v}$.

As for the first term of (6), since $\boldsymbol{X}^\top \boldsymbol{v} = \boldsymbol{0}$, consequently we get

$$\begin{aligned} \boldsymbol{w}^\top \boldsymbol{w} &= (\boldsymbol{X}\boldsymbol{\alpha} + \boldsymbol{v})^\top (\boldsymbol{X}\boldsymbol{\alpha} + \boldsymbol{v}) = \boldsymbol{\alpha}^\top \boldsymbol{X}^\top \boldsymbol{X}\boldsymbol{\alpha} + \boldsymbol{v}^\top \boldsymbol{v} \\ &\ge \boldsymbol{\alpha}^\top \boldsymbol{X}^\top \boldsymbol{X}\boldsymbol{\alpha} \end{aligned}$$

with equality occurring if and only if $\boldsymbol{v} = \boldsymbol{0}$.

So, setting $\boldsymbol{v} = \boldsymbol{0}$ does not affect the second, the third and the last term while strictly reduces the first term of (6). Hence, $\boldsymbol{w}^*$ for problem (6) admits a representation of the form (7). □

According to Theorem 1, we have

$$\begin{aligned} \boldsymbol{X}^\top \boldsymbol{w} &= \boldsymbol{X}^\top \boldsymbol{X}\boldsymbol{\alpha} = \boldsymbol{G}\boldsymbol{\alpha}, \\ \boldsymbol{w}^\top \boldsymbol{w} &= \boldsymbol{\alpha}^\top \boldsymbol{X}^\top \boldsymbol{X}\boldsymbol{\alpha} = \boldsymbol{\alpha}^\top \boldsymbol{G}\boldsymbol{\alpha}, \end{aligned}$$

where $\boldsymbol{G} = \boldsymbol{X}^\top \boldsymbol{X}$ is the kernel matrix. Let $\boldsymbol{G}_{:i}$ denote the i-th column of $\boldsymbol{G}$, then (6) can be cast as

$$\begin{aligned} \min_{\boldsymbol{\alpha}, \boldsymbol{\xi}} \quad & \frac{1}{2}\boldsymbol{\alpha}^\top \boldsymbol{Q}\boldsymbol{\alpha} + \boldsymbol{p}^\top \boldsymbol{\alpha} + C\sum_{i=1}^{m} \xi_i \\ \text{s.t.} \quad & y_i \boldsymbol{\alpha}^\top \boldsymbol{G}_{:i} \ge 1 - \xi_i, \\ & \xi_i \ge 0,\ i = 1, \ldots, m, \end{aligned} \tag{8}$$

where $\boldsymbol{Q} = 4\lambda_1(m\boldsymbol{G}^\top \boldsymbol{G} - (\boldsymbol{G}\boldsymbol{y})(\boldsymbol{G}\boldsymbol{y})^\top)/m^2 + \boldsymbol{G}$ and $\boldsymbol{p} = -\lambda_2 \boldsymbol{G}\boldsymbol{y}/m$. By introducing the lagrange multipliers $\boldsymbol{\beta} = [\beta_1, \ldots, \beta_m]^\top$ and $\boldsymbol{\eta} = [\eta_1, \ldots, \eta_m]^\top$ for the first and the second constraints respectively, the Lagrangian of (8) leads to

$$\begin{aligned} L(\boldsymbol{\alpha}, \boldsymbol{\xi}, \boldsymbol{\beta}, \boldsymbol{\eta}) = & \frac{1}{2}\boldsymbol{\alpha}^\top \boldsymbol{Q}\boldsymbol{\alpha} + \boldsymbol{p}^\top \boldsymbol{\alpha} + C\sum_{i=1}^{m} \xi_i \\ & - \sum_{i=1}^{m} \beta_i (y_i \boldsymbol{\alpha}^\top \boldsymbol{G}_{:i} - 1 + \xi_i) - \sum_{i=1}^{m} \eta_i \xi_i. \end{aligned} \tag{9}$$

By setting the partial derivations of $\{\boldsymbol{\alpha}, \boldsymbol{\xi}\}$ to zero, we have

$$\frac{\partial L}{\partial \boldsymbol{\alpha}} = \boldsymbol{Q}\boldsymbol{\alpha} + \boldsymbol{p} - \sum_{i=1}^{m} \beta_i y_i \boldsymbol{G}_{:i}, \tag{10}$$

$$\frac{\partial L}{\partial \xi_i} = C - \beta_i - \eta_i = 0,\ i = 1, \ldots, m. \tag{11}$$

Algorithm 1 Kernel LDM

Input: Data set $\boldsymbol{X}$, λ_1, λ_2, C
Output: $\boldsymbol{\alpha}$
Initialize $\boldsymbol{\beta} = \boldsymbol{0}$, $\boldsymbol{\alpha} = \frac{\lambda_2}{m}\boldsymbol{Q}^{-1}\boldsymbol{G}\boldsymbol{y}$, $\boldsymbol{A} = \boldsymbol{Q}^{-1}\boldsymbol{G}\boldsymbol{Y}$, $h_{ii} = \boldsymbol{e}_i^\top \boldsymbol{Y}\boldsymbol{G}\boldsymbol{Q}^{-1}\boldsymbol{G}\boldsymbol{Y}\boldsymbol{e}_i$;
while $\boldsymbol{\beta}$ not converge **do**
  for $i = 1, \ldots m$ **do**
    $[\nabla f(\boldsymbol{\beta})]_i \leftarrow \boldsymbol{e}_i^\top \boldsymbol{Y}\boldsymbol{G}\boldsymbol{\alpha} - 1$;
    $\beta_i^{old} \leftarrow \beta_i$;
    $\beta_i \leftarrow \min\left(\max\left(\beta_i - \frac{[\nabla f(\boldsymbol{\beta})]_i}{h_{ii}}, 0\right), C\right)$;
    $\boldsymbol{\alpha} \leftarrow \boldsymbol{\alpha} + (\beta_i - \beta_i^{old})\boldsymbol{A}\boldsymbol{e}_i$;
  end for
end while

By substituting (10) and (11) into (9), the dual [1] of (8) can be cast as:

$$\begin{aligned} \min_{\boldsymbol{\beta}} \quad & f(\boldsymbol{\beta}) = \frac{1}{2}\boldsymbol{\beta}^\top \boldsymbol{H}\boldsymbol{\beta} + \left(\frac{\lambda_2}{m}\boldsymbol{H}\boldsymbol{e} - \boldsymbol{e}\right)^\top \boldsymbol{\beta}, \\ \text{s.t.} \quad & 0 \le \beta_i \le C,\ i = 1, \ldots, m. \end{aligned} \tag{12}$$

where $\boldsymbol{H} = \boldsymbol{Y}\boldsymbol{G}\boldsymbol{Q}^{-1}\boldsymbol{G}\boldsymbol{Y}$, $\boldsymbol{Q}^{-1}$ refers to the inverse matrix of $\boldsymbol{Q}$ and $\boldsymbol{e}$ stands for the all-one vector. Due to the simple decoupled box constraint and the convex quadratic objective function, as suggested by [29], (12) can be efficiently solved by the dual coordinate descent method. In dual coordinate descent method [13], 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 β_i by keeping the other $\beta_{j \ne i}$'s as constants, one needs to solve the following subproblem,

$$\begin{aligned} \min_{t} \quad & f(\boldsymbol{\beta} + t\boldsymbol{e}_i) \\ \text{s.t.} \quad & 0 \le \beta_i + t \le C, \end{aligned} \tag{13}$$

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

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

where $[\nabla f(\boldsymbol{\beta})]_i$ is the i-th component of the gradient $\nabla f(\boldsymbol{\beta})$. Note that $f(\boldsymbol{\beta})$ is independent of t and thus can be dropped. Considering that $f(\boldsymbol{\beta} + t\boldsymbol{e}_i)$ is a simple quadratic function of t, and further note the box constraint $0 \le \alpha_i \le C$, the minimizer of (13) leads to a closed-form solution,

$$\beta_i^{new} = \min\left(\max\left(\beta_i - \frac{[\nabla f(\boldsymbol{\beta})]_i}{h_{ii}}, 0\right), C\right).$$

Algorithm 1 summarizes the pseudo-code of kernel LDM.

For prediction, according to (10), one can obtain the coefficients $\boldsymbol{\alpha}$ from the optimal $\boldsymbol{\beta}^*$ as

$$\begin{aligned} \boldsymbol{\alpha} &= \boldsymbol{Q}^{-1}(\boldsymbol{G}\boldsymbol{Y}\boldsymbol{\beta}^* - \boldsymbol{p}) = \boldsymbol{Q}^{-1}\left(\frac{\lambda_2}{m}\boldsymbol{G}\boldsymbol{Y}\boldsymbol{e} + \boldsymbol{G}\boldsymbol{Y}\boldsymbol{\beta}^*\right) \\ &= \boldsymbol{Q}^{-1}\boldsymbol{G}\boldsymbol{Y}\left(\frac{\lambda_2}{m}\boldsymbol{e} + \boldsymbol{\beta}^*\right). \end{aligned}$$

[1] Here we omit constants without influence on optimization.

Hence for testing instance $\boldsymbol{z}$, its label can be obtained by

$$sgn\left(\boldsymbol{w}^\top \phi(\boldsymbol{z})\right) = sgn\left(\sum_{i=1}^{m} \alpha_i k(\boldsymbol{x}_i, \boldsymbol{z})\right).$$

3.2.2 Large Scale Kernel LDM

In section 3.2.1, the proposed method can efficiently deal with kernel LDM. However, the inherent computational cost for the kernel matrix in kernel LDM takes $O(m^2)$ time, which might be computational prohibitive for large scale problems. To make LDM more useful, in the following, we present a fast linear kernel LDM for large scale problems by adopting the average stochastic gradient descent (ASGD) method [19].

For linear kernel LDM, (5) can be reformulated as the following form,

$$\begin{aligned}\min_{\boldsymbol{w}} \ g(\boldsymbol{w}) = &\frac{1}{2}\boldsymbol{w}^\top \boldsymbol{w} + \frac{2\lambda_1}{m^2}\boldsymbol{w}^\top (m\boldsymbol{X}\boldsymbol{X}^\top - \boldsymbol{X}\boldsymbol{y}\boldsymbol{y}^\top \boldsymbol{X}^\top)\boldsymbol{w} \\ &- \frac{\lambda_2}{m}(\boldsymbol{X}\boldsymbol{y})^\top \boldsymbol{w} + C\sum_{i=1}^{m} \max\{0, 1 - y_i \boldsymbol{w}^\top \boldsymbol{x}_i\}, \end{aligned} \quad (14)$$

where $\boldsymbol{X} = [\boldsymbol{x}_1 \dots \boldsymbol{x}_m]$, $\boldsymbol{y} = [y_1, \dots, y_m]^\top$ is a column vector.

For large scale problems, computing the gradient of (14) is expensive because its computation involves all the training examples. Stochastic gradient descent (SGD) works by computing a noisy unbiased estimation of the gradient via sampling a subset of the training examples. Theoretically, when the objective is convex, it can be shown that in expectation, SGD converges to the global optimal solution [15, 3]. During the past decade, SGD has been applied to various machine learning problems and achieved promising performances [30, 23, 2, 24].

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

THEOREM 2. *If two examples $(\boldsymbol{x}_i, y_i)$ and $(\boldsymbol{x}_j, y_j)$ are sampled from training set randomly, then*

$$\begin{aligned}\nabla g(\boldsymbol{w}, &\boldsymbol{x}_i, \boldsymbol{x}_j) = 4\lambda_1 \boldsymbol{x}_i \boldsymbol{x}_i^\top \boldsymbol{w} - 4\lambda_1 y_i \boldsymbol{x}_i y_j \boldsymbol{x}_j^\top \boldsymbol{w} + \boldsymbol{w} \\ &- \lambda_2 y_i \boldsymbol{x}_i - mC \begin{cases} y_i \boldsymbol{x}_i & i \in \boldsymbol{I}, \\ 0 & otherwise, \end{cases} \end{aligned} \quad (15)$$

where $\boldsymbol{I} \equiv \{i \mid y_i \boldsymbol{w}^\top \boldsymbol{x}_i < 1\}$ is an unbiased estimation of $\nabla g(\boldsymbol{w})$.

PROOF. Note that the gradient of $g(\boldsymbol{w})$ is

$$\nabla g(\boldsymbol{w}) = \boldsymbol{Q}\boldsymbol{w} + \boldsymbol{p} - C\sum_{i=1}^{m} y_i \boldsymbol{x}_i, i \in \boldsymbol{I},$$

where $\boldsymbol{Q} = 4\lambda_1(m\boldsymbol{X}\boldsymbol{X}^\top - \boldsymbol{X}\boldsymbol{y}(\boldsymbol{X}\boldsymbol{y})^\top)/m^2 + \boldsymbol{I}$ and $\boldsymbol{p} = -\lambda_2 \boldsymbol{X}\boldsymbol{y}/m$. Further note that

$$\begin{aligned} E_{\boldsymbol{x}_i}[\boldsymbol{x}_i \boldsymbol{x}_i^\top] &= \frac{1}{m}\sum_{i=1}^{m} \boldsymbol{x}_i \boldsymbol{x}_i^\top = \frac{1}{m}\boldsymbol{X}\boldsymbol{X}^\top, \\ E_{\boldsymbol{x}_i}[y_i \boldsymbol{x}_i] &= \frac{1}{m}\sum_{i=1}^{m} y_i \boldsymbol{x}_i = \frac{1}{m}\boldsymbol{X}\boldsymbol{y}. \end{aligned} \quad (16)$$

According to the linearity of expectation, the independence between $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$, and with (16), we have

$$\begin{aligned} &E_{\boldsymbol{x}_i, \boldsymbol{x}_j}[\nabla g(\boldsymbol{w}, \boldsymbol{x}_i, \boldsymbol{x}_j)] \\ &= 4\lambda_1 E_{\boldsymbol{x}_i}[\boldsymbol{x}_i \boldsymbol{x}_i^\top]\boldsymbol{w} - 4\lambda_1 E_{\boldsymbol{x}_i}[y_i \boldsymbol{x}_i] E_{\boldsymbol{x}_j}[y_j \boldsymbol{x}_j]^\top \boldsymbol{w} + \boldsymbol{w} \\ &\quad - \lambda_2 E_{\boldsymbol{x}_i}[y_i \boldsymbol{x}_i] - mC E_{\boldsymbol{x}_i}[y_i \boldsymbol{x}_i \mid i \in \boldsymbol{I}] \\ &= 4\lambda_1 \frac{1}{m}\boldsymbol{X}\boldsymbol{X}^\top \boldsymbol{w} - 4\lambda_1 \frac{1}{m}\boldsymbol{X}\boldsymbol{y}\left(\frac{1}{m}\boldsymbol{X}\boldsymbol{y}\right)^\top \boldsymbol{w} + \boldsymbol{w} \\ &\quad - \lambda_2 \frac{1}{m}\boldsymbol{X}\boldsymbol{y} - mC\frac{1}{m}\sum_{i=1}^{m} y_i \boldsymbol{x}_i, i \in \boldsymbol{I} \\ &= \boldsymbol{Q}\boldsymbol{w} + \boldsymbol{p} - C\sum_{i=1}^{m} y_i \boldsymbol{x}_i, i \in \boldsymbol{I} \\ &= \nabla g(\boldsymbol{w}). \end{aligned}$$

It is shown that $\nabla g(\boldsymbol{w}, \boldsymbol{x}_i, \boldsymbol{x}_j)$ is a noisy unbiased gradient of $g(\boldsymbol{w})$. □

Algorithm 2 Large Scale Kernel LDM

Input: Data set $\boldsymbol{X}$, λ_1, λ_2, C
Output: $\bar{\boldsymbol{w}}$
Initialize $\boldsymbol{u} = \boldsymbol{0}$, $T = 5$;
for $t = 1, \dots Tm$ **do**
  Randomly sample two training examples $(\boldsymbol{x}_i, y_i)$ and $(\boldsymbol{x}_j, y_j)$;
  Compute $\nabla g(\boldsymbol{w}, \boldsymbol{x}_i, \boldsymbol{x}_j)$ as in (15);
  $\boldsymbol{w} \leftarrow \boldsymbol{w} - \eta_t \nabla g(\boldsymbol{w}, \boldsymbol{x}_i, \boldsymbol{x}_j)$;
  $\bar{\boldsymbol{w}} \leftarrow \bar{\boldsymbol{w}} + \mu_t(\boldsymbol{w} - \bar{\boldsymbol{w}})$;
end for

With Theorem 2, the stochastic gradient update can be formed as

$$\boldsymbol{w}_{t+1} = \boldsymbol{w}_t - \eta_t \nabla g(\boldsymbol{w}, \boldsymbol{x}_i, \boldsymbol{x}_j), \quad (17)$$

where η_t is a suitably chosen step-size parameter in the t-th iteration.

In practice, we use averaged stochastic gradient descent (ASGD) which is more robust than SGD [28]. At each iteration, besides performing the normal stochastic gradient update (17), we also compute

$$\bar{\boldsymbol{w}}_t = \frac{1}{t - t_0}\sum_{i=t_0+1}^{t} \boldsymbol{w}_i,$$

where t_0 determines when we engage the averaging process. This average can be computed efficiently using a recursive formula:

$$\bar{\boldsymbol{w}}_{t+1} = \bar{\boldsymbol{w}}_t + \mu_t(\boldsymbol{w}_{t+1} - \bar{\boldsymbol{w}}_t),$$

where $\mu_t = 1/\max\{1, t - t_0\}$.

Algorithm 2 summarizes the pseudo-code of large scale kernel LDM.

3.3 Analysis

In this section, we study the statistical property of LDM. Specifically, we derive a bound on the expectation of error for LDM according to the leave-one-out cross-validation estimate, which is an unbiased estimate of the probability of test error.

Here we only consider the linear case (14) for simplicity, however, the results are also applicable to any other feature

mapping ϕ. Following the same steps in Section 3.2.1, one can have the dual problem of (14), i.e.,

$$\begin{aligned} \min_{\boldsymbol{\alpha}} \ & f(\boldsymbol{\alpha}) = \frac{1}{2}\boldsymbol{\alpha}^\top \boldsymbol{H}\boldsymbol{\alpha} + \left(\frac{\lambda_2}{m}\boldsymbol{H}\boldsymbol{e} - \boldsymbol{e}\right)^\top \boldsymbol{\alpha}, \\ \text{s.t.} \ & 0 \le \alpha_i \le C, \ i = 1, \ldots, m. \end{aligned} \tag{18}$$

where $\boldsymbol{H} = \boldsymbol{Y}\boldsymbol{X}^\top \boldsymbol{Q}^{-1}\boldsymbol{X}\boldsymbol{Y}$, $\boldsymbol{Q} = \frac{4\lambda_1}{m^2}(m\boldsymbol{X}\boldsymbol{X}^\top - \boldsymbol{X}\boldsymbol{y}(\boldsymbol{X}\boldsymbol{y})^\top) + \boldsymbol{I}$, $\boldsymbol{e}$ stands for the all-one vector and $\boldsymbol{Q}^{-1}$ refers to the inverse matrix of $\boldsymbol{Q}$.

THEOREM 3. *Let $\boldsymbol{\alpha}$ denote the optimal solution of (18), and $E[R(\boldsymbol{\alpha})]$ be the expectation of the probability of error, then we have*

$$E[R(\boldsymbol{\alpha})] \le \frac{E[h\sum_{i\in \boldsymbol{I}_1}\alpha_i + |\boldsymbol{I}_2|]}{m}, \tag{19}$$

where $\boldsymbol{I}_1 \equiv \{i \mid 0 < \alpha_i < C\}$, $\boldsymbol{I}_2 \equiv \{i \mid \alpha_i = C\}$ and $h = \max\{diag\{\boldsymbol{H}\}\}$.

PROOF. Suppose

$$\begin{aligned} \boldsymbol{\alpha}^* &= \operatorname*{argmin}_{0\le\boldsymbol{\alpha}\le C} f(\boldsymbol{\alpha}), \\ \boldsymbol{\alpha}^i &= \operatorname*{argmin}_{0\le\boldsymbol{\alpha}\le C, \alpha_i = 0} f(\boldsymbol{\alpha}), \quad i = 1, \ldots, m, \end{aligned} \tag{20}$$

and the corresponding solution for the linear kernel LDM are $\boldsymbol{w}^*$ and $\boldsymbol{w}^i$, respectively.

As shown in [17],

$$E[R(\boldsymbol{\alpha})] = \frac{E[\mathcal{L}((\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_m, y_m))]}{m}, \tag{21}$$

where $\mathcal{L}((\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_m, y_m))$ is the number of errors in the leave-one-out procedure. Note that if $\alpha_i^* = 0$, $(\boldsymbol{x}_i, y_i)$ will always be classified correctly in the leave-one-out procedure according to the KKT conditions. So for any misclassified example $(\boldsymbol{x}_i, y_i)$, we only need to consider the following two cases:

1) $0 < \alpha_i^* < C$, according to the definition in (20), we have

$$f(\boldsymbol{\alpha}^i) - \min_t f(\boldsymbol{\alpha}^i + t\boldsymbol{e}_i) \le f(\boldsymbol{\alpha}^i) - f(\boldsymbol{\alpha}^*), \tag{22}$$

$$f(\boldsymbol{\alpha}^i) - f(\boldsymbol{\alpha}^*) \le f(\boldsymbol{\alpha}^* - \alpha_i^*\boldsymbol{e}_i) - f(\boldsymbol{\alpha}^*), \tag{23}$$

where $\boldsymbol{e}_i$ denotes a vector with 1 in the i-th coordinate and 0's elsewhere. We can find that, the left-hand side of (22) is equal to $(1 - y_i\boldsymbol{x}_i^\top \boldsymbol{w}^i)^2/2h_{ii}$, and the right-hand side of (23) is equal to $\alpha_i^{*2}h_{ii}/2$. So by combining (22) and (23), we have

$$(1 - y_i\boldsymbol{x}_i^\top \boldsymbol{w}^i)^2/2h_{ii} \le \alpha_i^{*2}h_{ii}/2.$$

Further note that $y_i\boldsymbol{x}_i^\top \boldsymbol{w}^i < 0$, rearranging the above we can obtain $1 \le \alpha_i^* h_{ii}$.

2) $\alpha_i^* = C$, all these examples will be misclassified in the leave-one-out procedure.

So we have

$$\mathcal{L}((\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_m, y_m)) \le h\sum_{i\in\boldsymbol{I}_1}\alpha_i^* + |\boldsymbol{I}_2|,$$

where $\boldsymbol{I}_1 \equiv \{i \mid 0 < \alpha_i^* < C\}$, $\boldsymbol{I}_2 \equiv \{i \mid \alpha_i^* = C\}$ and $h = \max\{h_{ii}, i = 1, \ldots, m\}$. Take expectation on both side and with (21), we get that (19) holds. □

4. EMPIRICAL STUDY

In this section, we empirically evaluate the effectiveness of LDM on a broad range of data sets. We first introduce the experimental settings in Section 4.1, and then compare LDM with SVM and three state-of-the-art approaches[2] in Section 4.2 and Section 4.3. In addition, we also study the cumulative margin distribution produced by LDM and SVM in Section 4.4. The computational cost and parameter influence are presented in Section 4.5 and Section 4.6, respectively.

4.1 Experimental Setup

We evaluate the effectiveness of our proposed LDMs on twenty regular scale data sets and twelve large scale data sets, including both UCI data sets and real-world data sets like KDD2010[3]. Table 1 summarizes the statistics of these data sets. The data set size is ranged from 106 to more than 8,000,000, and the dimensionality is ranged from 2 to more than 20,000,000, covering a broad range of properties. All features are normalized into the interval $[0, 1]$. For each data set, half of examples are randomly selected as the training data, and the remaining examples are used as the testing data. For regular scale data sets, both linear and RBF kernels are evaluated. Experiments are repeated for 30 times with random data partitions, and the average accuracies as well as the standard deviations are recorded. For large scale data sets, linear kernel is evaluated. Experiments are repeated for 10 times with random data partitions, and the average accuracies (with standard deviations) are recorded.

LDMs are compared with standard SVMs which ignore the margin distribution, and three state-of-the-art methods, that is, Margin Distribution Optimization (MDO) [12], Maximal Average Margin for Classifiers (MAMC) [18] and Kernel Method for the direct Optimization of the Margin Distribution (KM-OMD) [1]. For SVM, KM-OMD and LDM, the regularization parameter C is selected by 5-fold cross validation from $[10, 50, 100]$. For MDO, the parameters are set as the recommended parameters in [12]. For LDM, the regularization parameters λ_1, λ_2 are selected by 5-fold cross validation from the set of $[2^{-8}, \ldots, 2^{-2}]$, the parameters η_t and t_0 are set with the same setup in [28], and T is fixed to 5. The width of the RBF kernel for SVM, MAMC, KM-OMD and LDM are selected by 5-fold cross validation from the set of $[2^{-2}\delta, \ldots, 2^2\delta]$, where δ is the average distance between instances. All selections are performed on training sets.

4.2 Results on Regular Scale Data Sets

Tables 2 and 3 summarize the results on twenty regular scale data sets. As can be seen, the overall performance of LDM is superior or highly competitive to SVM and other compared methods. Specifically, for linear kernel, LDM performs significantly better than SVM, MDO, MAMC, KM-OMD on 12, 9, 17 and 10 over 20 data sets, respectively, and achieves the best accuracy on 13 data sets; for RBF kernel, LDM performs significantly better than SVM, MAMC, KM-OMD on 10, 18 and 15 over 20 data sets, respectively, and achieves the best accuracy on 15 data sets. MDO is not compared since it is specified for the linear kernel. In addition, as can be seen, in comparing with standard SVM which does not consider margin distribution,

[2]These approaches will be briefly introduced in Section 5.
[3]https://pslcdatashop.web.cmu.edu/KDDCup/downloads.jsp

Table 1: Characteristics of experimental data sets.

Scale	Dataset	#Instance	#Feature	Dataset	#Instance	#Feature
regular	*promoters*	106	57	*haberman*	306	14
	planning	182	12	*vehicle*	435	16
	colic	188	13	*clean1*	476	166
	parkinsons	195	22	*wdbc*	569	14
	colic.ORIG	205	17	*isolet*	600	51
	sonar	208	60	*credit-a*	653	15
	vote	232	16	*austra*	690	15
	house	232	16	*australian*	690	42
	heart	270	9	*fourclass*	862	2
	breast	277	9	*german*	1,000	59
large	*farm-ads*	4,143	54,877	*ijcnn1*	141,691	22
	news20	19,996	1,355,191	*skin*	245,057	3
	adult-a	32,561	123	*covtype*	581,012	54
	w8a	49,749	300	*rcv1*	697,641	47,236
	cod-rna	59,535	8	*url*	2,396,130	3,231,961
	real-sim	72,309	20,958	*kdd2010*	8,407,752	20,216,830

Table 2: Accuracy (mean±std.) comparison on regular scale data sets. Linear kernels are used. The best accuracy on each data set is bolded. •/○ indicates the performance is significantly better/worse than SVM (paired t-tests at 95% significance level). The win/tie/loss counts are summarized in the last row.

Dataset	SVM	MDO	MAMC	KM-OMD	LDM
promoters	0.723±0.071	0.713±0.067	0.520±0.096○	**0.736±0.061**	0.721±0.069
planning-relax	0.683±0.031	0.605±0.185○	**0.706±0.034**•	0.479±0.050○	**0.706±0.034**•
colic	0.814±0.035	0.781±0.154	0.661±0.062○	0.813±0.028	**0.832±0.026**•
parkinsons	0.846±0.038	0.732±0.270○	0.764±0.035○	0.814±0.024○	**0.865±0.030**•
colic.ORIG	0.618±0.027	0.624±0.040	0.623±0.027	**0.635±0.045**•	0.619±0.042
sonar	0.725±0.039	0.734±0.035	0.533±0.045○	**0.766±0.033**•	0.736±0.036
vote	0.934±0.022	0.587±0.435○	0.884±0.022○	0.957±0.013•	**0.970±0.014**•
house	0.942±0.015	0.943±0.015	0.883±0.029○	0.957±0.020•	**0.968±0.011**•
heart	0.799±0.029	0.826±0.026•	0.537±0.057○	**0.836±0.026**•	0.791±0.030
breast-cancer	0.717±0.033	0.710±0.031	0.706±0.027	0.696±0.031○	**0.725±0.027**•
haberman	0.734±0.030	0.728±0.029	**0.738±0.020**	0.667±0.040○	**0.738±0.020**
vehicle	0.959±0.012	0.956±0.012	0.566±0.160○	**0.960±0.010**	0.959±0.013
clean1	0.803±0.035	0.798±0.031	0.561±0.025○	**0.821±0.027**•	0.814±0.019•
wdbc	0.963±0.012	0.966±0.010	0.623±0.020○	0.968±0.009•	**0.968±0.011**•
isolet	0.995±0.003	0.501±0.503○	0.621±0.207○	0.995±0.003	**0.997±0.002**•
credit-a	0.861±0.014	0.862±0.013	0.596±0.063○	0.863±0.013	**0.864±0.013**•
austra	0.857±0.013	0.842±0.055	0.567±0.044○	0.858±0.013	**0.859±0.015**
australian	0.844±0.019	0.842±0.020	0.576±0.049○	0.858±0.016•	**0.866±0.014**•
fourclass	0.724±0.014	0.377±0.238○	0.641±0.020○	**0.736±0.014**•	0.723±0.014
german	0.711±0.030	0.737±0.014•	0.697±0.017○	0.729±0.017•	**0.738±0.016**•
Ave. accuracy	0.813	0.743	0.650	0.807	0.823
LDM: w/t/l	12/8/0	9/10/1	17/3/0	10/5/5	

the win/tie/loss counts show that LDM is always better or comparable, never worse than SVM.

4.3 Results on Large Scale Data Sets

Table 4 summarizes the results on twelve large scale data sets. KM-OMD did not return results on all data sets and MDO did not return results on KDD2010 in 48 hours due to the high computational cost. As can be seen, the overall performance of LDM is superior or highly competitive to SVM and other compared methods. Specifically, LDM performs significantly better than SVM, MDO, MAMC on 6, 7 and 12 over 12 data sets, respectively, and achieves the best accuracy on 8 data sets. In addition, the win/tie/loss counts show that LDM is always better or comparable, never worse than SVM.

4.4 Margin Distributions

Figure 1 plots the cumulative margin distribution of SVM and LDM on some representative regular scale data sets. The curves for other data sets are similar. The point where a curve and the x-axis crosses is the corresponding minimum margin. As can be seen, LDM usually has a little bit smaller minimum margin than SVM, whereas the LDM curve generally lies on the right side, showing that the

Table 3: Accuracy (mean±std.) comparison on regular scale data sets. RBF kernels are used. The best accuracy on each data set is bolded. •/○ indicates the performance is significantly better/worse than SVM (paired t-tests at 95% significance level). The win/tie/loss counts are summarized in the last row. MDO does not have results since it is specified for the linear kernel.

Dataset	SVM	MDO	MAMC	KM-OMD	LDM
promoters	0.684±0.100	N/A	0.638±0.121○	0.701±0.085	**0.715±0.074•**
planning-relax	**0.708±0.035**	N/A	0.706±0.034	0.683±0.031○	0.707±0.034
colic	0.822±0.033	N/A	0.623±0.037○	0.825±0.024	**0.841±0.018•**
parkinsons	**0.929±0.029**	N/A	0.852±0.036○	0.906±0.033○	0.927±0.029
colic.ORIG	0.638±0.043	N/A	0.623±0.027	0.621±0.039	**0.641±0.044**
sonar	0.842±0.034	N/A	0.753±0.052○	0.821±0.051○	**0.846±0.032**
vote	0.946±0.016	N/A	0.913±0.019○	0.930±0.029○	**0.968±0.013•**
house	0.953±0.020	N/A	0.561±0.139○	0.938±0.022○	**0.964±0.013•**
heart	0.808±0.025	N/A	0.540±0.043○	0.805±0.048	**0.822±0.029•**
breast-cancer	0.729±0.030	N/A	0.706±0.027○	0.691±0.024○	**0.753±0.027•**
haberman	0.727±0.024	N/A	**0.742±0.021•**	0.676±0.042○	0.731±0.027
vehicle	0.992±0.007	N/A	0.924±0.025○	0.988±0.008○	**0.993±0.006**
clean1	0.890±0.020	N/A	0.561±0.025○	0.772±0.043○	**0.891±0.024**
wdbc	0.951±0.011	N/A	0.740±0.042○	0.941±0.040	**0.961±0.010•**
isolet	0.998±0.002	N/A	0.994±0.004○	0.995±0.003○	**0.998±0.002**
credit-a	0.858±0.014	N/A	0.542±0.032○	0.845±0.029○	**0.861±0.013**
austra	0.853±0.013	N/A	0.560±0.018○	0.854±0.017	**0.857±0.014•**
australian	0.815±0.014	N/A	0.554±0.015○	**0.860±0.014•**	0.854±0.016•
fourclass	**0.998±0.003**	N/A	0.791±0.014○	0.838±0.014○	0.998±0.003
german	0.731±0.019	N/A	0.697±0.017○	0.742±0.017•	**0.743±0.016•**
Ave. accuracy	0.844	N/A	0.701	0.822	0.854
LDM: w/t/l	10/10/0	N/A	18/1/1	15/5/0	

Table 4: Accuracy (mean±std.) comparison on large scale data sets. Linear kernels are used. The best accuracy on each data set is bolded. •/○ indicates the performance is significantly better/worse than SVM (paired t-tests at 95% significance level). The win/tie/loss counts are summarized in the last row. KM-OMD and MDO did not return results on some data sets in 48 hours.

Dataset	SVM	MDO	MAMC	KM-OMD	LDM
farm-ads	0.880±0.007	0.880±0.007	0.759±0.038○	N/A	**0.890±0.008•**
news20	0.954±0.002	0.948±0.002○	0.772±0.017○	N/A	**0.960±0.001•**
adult-a	0.845±0.002	0.788±0.053○	0.759±0.002○	N/A	**0.846±0.003•**
w8a	0.983±0.001	**0.985±0.001•**	0.971±0.001○	N/A	0.983±0.001
cod-rna	**0.899±0.001**	0.774±0.203	0.667±0.001○	N/A	0.899±0.001
real-sim	0.961±0.001	0.955±0.002○	0.744±0.004○	N/A	**0.971±0.001•**
ijcnn1	0.921±0.003	**0.921±0.002**	0.904±0.001○	N/A	0.921±0.002
skin	0.934±0.001	0.929±0.003○	0.792±0.000○	N/A	**0.934±0.001**
covtype	0.762±0.001	0.760±0.003○	0.628±0.002○	N/A	**0.763±0.001**
rcv1	0.969±0.000	0.959±0.000○	0.913±0.000○	N/A	**0.977±0.000•**
url	**0.993±0.006**	0.993±0.006	0.670±0.000○	N/A	0.993±0.006
kdd2010	0.852±0.001	N/A	0.853±0.000•	N/A	**0.881±0.001•**
Ave. accuracy	0.913	0.899	0.786	N/A	0.919
LDM: w/t/l	6/6/0	7/3/1	12/0/0	N/A	

margin distribution of LDM is generally better than that of SVM. In other words, for most examples, LDM generally produce a larger margin than SVM.

4.5 Time Cost

We compare the time cost of LDM and SVM on the twelve large scale data sets. All the experiments are performed with MATLAB 2012b on a machine with 8×2.60 GHz CPUs and 16GB main memory. The average CPU time (in seconds) on each data set is shown in Figure 2. We denote SVM implemented by the LIBLINEAR [9] package as SVM_l and SVM implemented by ASGD[4] as SVM_a, respectively. It can be seen that, both SVM_a and LDM are much faster than SVM_l, owing to the use of ASGD. LDM is just slightly slower than SVM_a on three data sets (news20, real-sim and skin) but highly competitive with SVM_a on the other nine data sets. Note that both SVM_l and SVM_a are very fast implementa-

[4]http://leon.bottou.org/projects/sgd

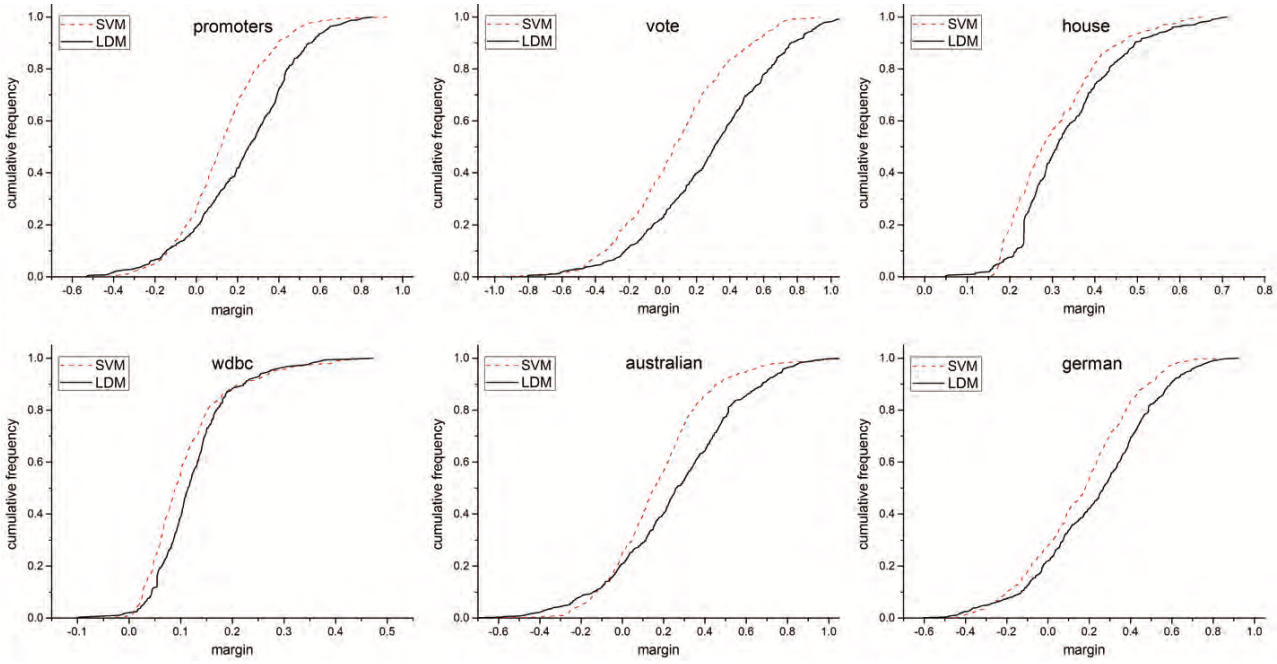

Figure 1: Cumulative frequency (y-axis) with respect to margin (x-axis) of SVM and LDM on some representative regular scale data sets. The more right the curve, the larger the accumulated margin.

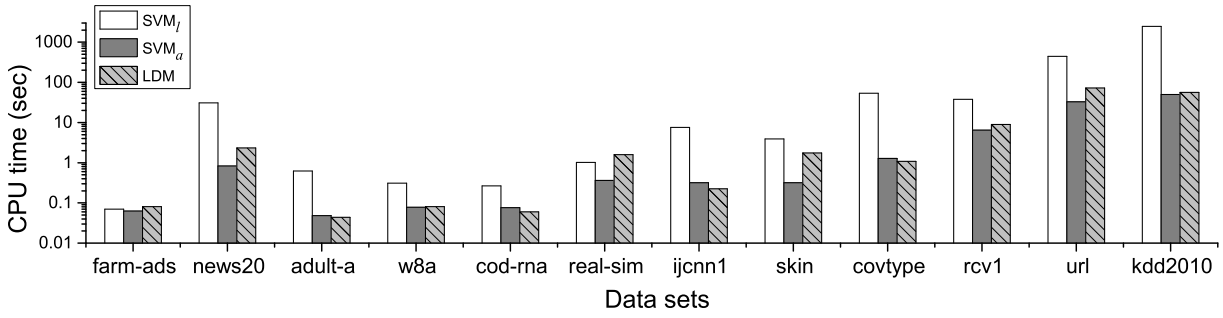

Figure 2: CPU time on the large scale data sets.

tions of SVMs; this shows that LDM is also computationally efficient.

4.6 Parameter Influence

LDM has three regularization parameters, i.e., λ_1, λ_2 and C. In previous empirical studies, they are set according to cross validation. Figure 3 further studies the influence of them on some representative regular scale data sets by fixing other parameters. Specifically, Figure 3(a) shows the influence of λ_1 on the accuracy by varying it from 2^{-8} to 2^{-2} while fixing λ_2 and C as the value suggested by the cross validation described in Section 4.1. Figure 3(b) and Figure 3(c) are obtained in the same way. It can be seen that, the performance of LDM is not very sensitive to the setting of the parameters, making LDM even more attractive in practice.

5. RELATED WORK

There are a few studies considered margin distribution in SVM-like algorithms [12, 18, 1]. Garg et al. [12] proposed the Margin Distribution Optimization (MDO) algorithm which minimizes the sum of the cost of each instance, where the cost is a function which assigns larger values to instances with smaller margins. MDO can be viewed as a method of optimizing weighted margin combination, where the weights are related to the margins. The objective function optimized by MDO, however, is non-convex, and thus, it may get stuck in local minima. In addition, MDO can only be used for linear kernel. As our experiments in Section 4 disclosed, the performance of MDO is inferior to LDM.

Pelckmans et al. [18] proposed the Maximal Average Margin for Classifiers (MAMC) and it can be viewed as a special case of LDM assuming that the margin variance is zero. MAMC has a closed-form solution, however, it will degenerate to a trivial solution when the classes are not with equal sizes. Our experiments in Section 4 showed that LDM is clearly superior to MAMC.

Aiolli et al. [1] proposed a Kernel Method for the direct Optimization of the Margin Distribution (KM-OMD) from a game theoretical perspective. Similar to MDO, this method

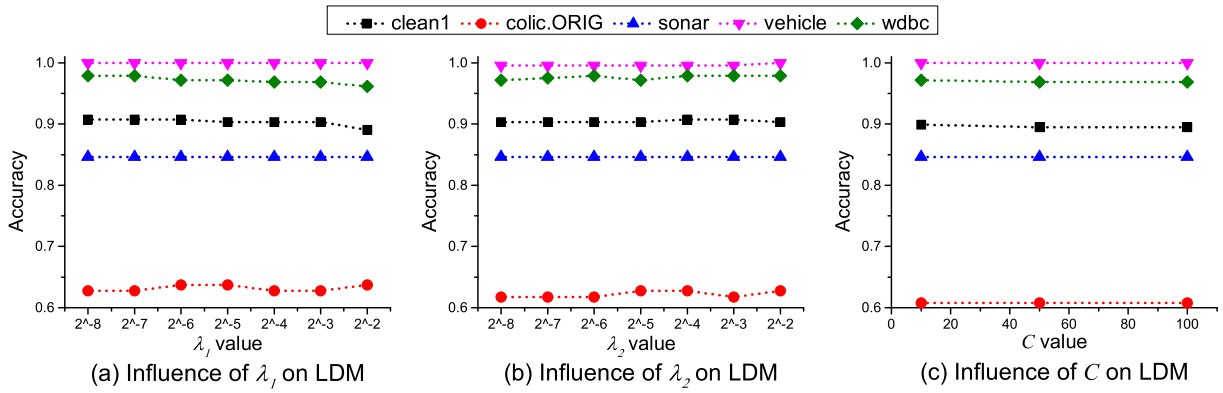

Figure 3: Parameter influence on some representative regular scale data sets.

also directly optimizes a weighted combination of margins over the training data, ignoring the influence of margin variances. Besides, this method considers hard-margin only, which may be another reason why it behaves worse than our method. It is noteworthy that the computational cost prohibits KM-OMD to be applied to large scale data, as shown in Table 4.

6. CONCLUSIONS

Support vector machines work by maximizing the minimum margin. Recent theoretical results suggested that the margin distribution, rather than a single-point margin such as the minimum margin, is more crucial to the generalization performance. In this paper, we propose the large margin distribution machine (LDM) which tries to optimize the margin distribution by maximizing the margin mean and minimizing the margin variance simultaneously. The LDM is a general learning approach which can be used in any place where SVM can be applied. Comprehensive experiments on twenty regular scale data sets and twelve large scale data sets validate the superiority of LDM to SVMs and many state-of-the-art methods. In the future it will be interesting to generalize the idea of LDM to regression and other learning settings.

7. ACKNOWLEDGMENTS

The authors want to thank anonymous reviewers for helpful comments and suggestions. This research was supported by the National Science Foundation of China (61333014) and the National Key Basic Research Program of China (2014CB340501).

8. REFERENCES

[1] F. Aiolli, G. San 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*, pages 305–314, Prague, Czech, 2008.

[2] A. Bordes, L. Bottou, and P. Gallinari. Sgd-qn: Careful quasi-newton stochastic gradient descent. *Journal of Machine Learning Research*, 10:1737–1754, 2009.

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

[4] L. Breiman. Prediction games and arcing classifiers. *Neural Computation*, 11(7):1493–1517, 1999.

[5] C. Cortes and V. Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, 1995.

[6] A. Cotter, S. Shalev-shwartz, and N. Srebro. Learning optimally sparse support vector machines. In *Proceedings of the 30th International Conference on Machine Learning*, pages 266–274, Atlanta, GA, 2013.

[7] N. Cristianini and J. Shawe-Taylor. *An Introduction to Support Vector Machines and Other Kernel-based Learning Methods*. Cambridge University Press, Cambridge, UK, 2000.

[8] H. Do and K. Alexandre. Convex formulations of radius-margin based support vector machines. In *Proceedings of the 30th International Conference on Machine Learning*, pages 169–177, Atlanta, GA, 2013.

[9] 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*, 9:1871–1874, 2008.

[10] 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*, pages 23–37, Barcelona, Spain, 1995.

[11] W. Gao and Z.-H. Zhou. On the doubt about margin explanation of boosting. *Artificial Intelligence*, 199–200:22–44, 2013.

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

[13] 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*, pages 408–415, Helsinki, Finland, 2008.

[14] C. Jose, P. Goyal, P. Aggrwal, and M. Varma. Local deep kernel learning for efficient non-linear svm prediction. In *Proceedings of the 30th International Conference on Machine Learning*, pages 486–494, Atlanta, GA, 2013.

[15] H. J. Kushner and G. G. Yin. *Stochastic approximation and recursive algorithms and applications; 2nd ed.* Springer, New York, 2003.

[16] S. Lacoste-julien, M. Jaggi, M. Schmidt, and P. Pletscher. Block-coordinate frank-wolfe optimization for structural svms. In *Proceedings of the 30th International Conference on Machine Learning*, pages 53–61, Atlanta, GA, 2013.

[17] A. Luntz and V. Brailovsky. On estimation of characters obtained in statistical procedure of recognition (in russian). *Technicheskaya Kibernetica*, 3, 1969.

[18] K. Pelckmans, J. Suykens, and B. D. Moor. A risk minimization principle for a class of parzen estimators. In J. Platt, D. Koller, Y. Singer, and S. Roweis, editors, *Advances in Neural Information Processing Systems 20*, pages 1137–1144. MIT Press, Cambridge, MA, 2008.

[19] B. T. Polyak and A. B. Juditsky. Acceleration of stochastic approximation by averaging. *SIAM Journal on Control and Optimization*, 30(4):838–855, 1992.

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

[21] R. E. Schapire, Y. Freund, P. L. Bartlett, and W. S. Lee. Boosting the margin: a new explanation for the effectives of voting methods. *Annuals of Statistics*, 26(5):1651–1686, 1998.

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

[23] S. Shalev-Shwartz, Y. Singer, and N. Srebro. Pegasos: Primal estimated sub-gradient solver for svm. In *Proceedings of the 24th International Conference on Machine Learning*, pages 807–814, Helsinki, Finland, 2007.

[24] O. Shamir and T. Zhang. Stochastic gradient descent for non-smooth optimization: Convergence results and optimal averaging schemes. In *Proceedings of the 30th International Conference on Machine Learning*, pages 71–79, Atlanta, GA, 2013.

[25] M. Takac, A. Bijral, P. Richtarik, and N. Srebro. Mini-batch primal and dual methods for svms. In *Proceedings of the 30th International Conference on Machine Learning*, pages 1022–1030, Atlanta, GA, 2013.

[26] V. Vapnik. *The Nature of Statistical Learning Theory*. Springer-Verlag, New York, 1995.

[27] L. W. Wang, M. Sugiyama, C. Yang, Z.-H. Zhou, and J. Feng. A refined margin analysis for boosting algorithms via equilibrium margin. *Journal of Machine Learning Research*, 12:1835–1863, 2011.

[28] W. Xu. Towards optimal one pass large scale learning with averaged stochastic gradient descent. *CoRR, abs/1107.2490*, 2011.

[29] G. X. Yuan, C. H. Ho, and C. J. Lin. Recent advances of large-scale linear classification. *Proceedings of the IEEE*, 100(9):2584–2603, 2012.

[30] T. Zhang. Solving large scale linear prediction problems using stochastic gradient descent algorithms. In *Proceedings of the 21st International Conference on Machine learning*, pages 116–123, Banff, Canada, 2004.

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