Sketch discriminatively regularized online gradient descent classification

Hui Xue & Zhen Ren

Applied Intelligence
The International Journal of Research on Intelligent Systems for Real Life Complex Problems

ISSN 0924-669X

Appl Intell
DOI 10.1007/s10489-019-01590-6

Springer

Applied Intelligence
https://doi.org/10.1007/s10489-019-01590-6

Sketch discriminatively regularized online gradient descent classification

Hui Xue[1,2] · Zhen Ren[1,2]

Abstract

Online learning represents an important family of efficient and scalable algorithms for large-scale classification problems. Many of them are linear with fast computational speed, but when faced with complex classification, they more likely have low accuracies. In order to improve accuracies, kernel trick is applied, however, it often brings high computational cost. In fact, discriminative information is vital in classification which is still not fully utilized in these algorithms. In this paper, we proposed a novel online linear method, called Sketch Discriminatively Regularized Online Gradient Descent Classification (SDROGD). In order to exploit inter-class separability and intra-class compactness, SDROGD utilizes a matrix to characterize the discriminative information and embeds it directly into a new regularization term. This matrix can be updated by the sketch technique in an online manner. After applying a simple but effective optimization, we show that SDROGD has a good time complexity bound, which is linear with the feature dimension or the number of samples. Experimental results on both toy and real-world datasets demonstrate that SDROGD has not only faster computational speed but also much better classification accuracies than some related kernelized algorithms.

Keywords Machine learning · Online learning · Classification · Sketch technique

1 Introduction

Online learning refers to a sequential machine learning task where a predictive model is learned incrementally from a sequence of data samples [1]. Unlike batch-wise methods, online methods often use one sample or a mini batch of samples per iteration to optimize their objective functions [2]. As a result, they require less computational cost and fewer memory resources and usually are used to deal with large-scale problems [3].

Over the past decades, a number of online linear methods have been proposed, including perceptron [1], online passive-aggressive algorithms (PA) [4], Pegasos [5], and so on. Perceptron tries to find a classification hyperplane that can separate the samples completely. Its optimizer is stochastic gradient descent (SGD). PA optimizes its objective function with the instantaneous loss from the new coming sample, and it has a closed-form updating equation [4]. Pegasos bases on offline support vector machine (SVM) problem directly and also uses SGD to optimize the objective function [5]. Different from PA and perceptron, Pegasos calculates the gradient and updates the parameters by a mini batch of samples at each iteration. It has been proved that Pegasos can converge to the actual SVM solution using a decayed learning rate [5]. Due to the linear characteristic, these methods have excellent computational speeds. However, their classification accuracies are relatively lower, especially in complex classification problems.

In order to improve the accuracies, a large number of kernelized methods have been presented. Kernelized perceptron (KP) is the dual form of perceptron using kernel trick [6]. Online gradient descent (OGD) can be considered as the kernelized version of Pegasos with the batch size of samples reducing to one [7]. Although they have better accuracies, both KP and OGD have to preserve all the mislabeled samples in memory. As a result, they suffer from the severe

✉ Hui Xue
hxue@seu.edu.cn

Zhen Ren
legend94rz@seu.edu.cn

1 School of Computer Science and Engineering, Southeast University, Nanjing, China

2 MOE Key Laboratory of Computer Network and Information Integration, Southeast University, Nanjing, China

Published online: 16 January 2020

computational and storage burden when faced with large-scale data [8]. To alleviate the resource burden of KP and OGD, budget online gradient descent (BOGD) is further proposed which utilizes some strategies to maintain a fixed number of mislabeled samples [9]. For example, it removes the oldest one when the budget overflows for adding a new one. Although fixing the number of mislabeled samples can decrease the resource cost, BOGD performs more poorly than KP and OGD due to the loss of much useful information. Fourier online gradient descent (FOGD) and Nyström online gradient descent (NOGD) aim to approximate the kernel matrix directly, which also avoid storing all the mislabeled samples. FOGD adopts random Fourier feature sampling technique, while NOGD uses Nyström method to approximate the matrix by a low-rank matrix [10]. Another branch of online methods is based on least square SVM, which obtains the solution by solving linear equations rather than a quadratic programming problem [11]. One of the latest related methods is called budget online least square SVM (BOLSSVM) [12]. Although it computes matrix inverse incrementally by constructing some vectors elaborately to alleviate the cost, it still has a budget for mislabeled samples. These methods introduce kernel, so they usually have better performance than linear ones. However, they suffer from expensive computational cost. Consequently, how to keep high classification accuracy and low computational burden simultaneously is vital for online learning.

In fact, discriminative information plays the most important role in classification which directly influences the hyperplane construction. However, most existing online methods usually use the discriminative information only in the loss function, that is, this information is not fully utilized. Considered that the primary goal of classification is to separate the samples of different classes in the output space as far as possible, embedding discriminative information into the regularization term could model this metric.

In this paper, we propose a novel online linear method, termed as Sketch Discriminatively Regularized Online Gradient Descent Classification (SDROGD). To emphasize on the discriminability of the hyperplane, SDROGD uses a discriminative information matrix to describe the corresponding information involved in the samples and embeds it into a new discriminative regularization term. This new term can effectively reflect inter-class separability and intra-class compactness in an online manner. An approximate approach, sketch technique, is used to update the matrix. However, the sketch technique needs to maintain the matrix explicitly in memory, which may still lead to expensive cost. Therefore, a targeted optimization is adopted to cut down the computational and storage cost of SDROGD, especially on high-dimensional datasets. Theoretical analyses about the approximative property of sketch technique, the convergence property, and the computational complexity are also given. Experimental results show that SDROGD possesses not only faster computational speed but also higher classification accuracies even than some kernelized methods. We also test the effect of some hyperparameters, which shows that SDROGD is insensitive about them to some degree.

In summary, the contributions of this paper are that we proposed a new online linear classification method, called SDROGD, to deal with the large-scale dataset. SDROGD is a linear method and it doesn't store any mislabeled sample, so it has good time complexity bound. To our best knowledge, this is the first online classifier embeds discriminative information into regularization term. Besides, it is possible to extend SDROGD to deal with multi-class problems.

The rest of this paper is organized as follows. Section 2 gives an overview of some related works. In Section 3, we present our SDROGD model and its optimization. The corresponding theoretical analysis is conducted in Section 4. Experimental setting and results are shown in Section 5. After giving the possible extension for multi-class in Section 6, we conclude our work in Section 7 finally.

2 Related work

Pegasos is an online linear classifier which is based on the offline SVM problem. It uses SGD to optimize its objective function rather than sequential minimal optimization (SMO), and thus has excellent computational speed. Given the training set $S = \{(\mathbf{x}_i, y_i)\}_{i=1}^{N}$, where $\mathbf{x}_i \in \mathbb{R}^{d\times 1}$, $y_i \in \{-1, 1\}$ and d is the number of feature dimension, the offline SVM problem is:

$$f_p(\mathbf{w}; S) = \frac{1}{|S|}\sum_{(\mathbf{x},y)\in S} l(\mathbf{w}; (\mathbf{x}, y)) + \frac{\lambda}{2}\mathbf{w}^T\mathbf{w} \tag{1}$$

where $l(\mathbf{w}; (\mathbf{x}, y))$ is the hinge loss function defined as follows:

$$l(\mathbf{w}; (\mathbf{x}, y)) = \max(0, 1 - y(\mathbf{w}^T\mathbf{x})) \tag{2}$$

$\mathbf{w}^T\mathbf{w}$ is the common Tikhonov regularization term. $\lambda > 0$.

Its optimizer SGD works iteratively. At iteration t, Pegasos firstly selects a mini batch of samples A_t randomly and then uses the gradient of $f_p(\mathbf{w}; A_t)$ to update its parameters $\mathbf{w}$.

The updating for $\mathbf{w}$ is:

$$\begin{aligned}\mathbf{w}_t &= \mathbf{w}_{t-1} - \alpha\nabla f_p(\mathbf{w}; A_t)\\ &= \mathbf{w}_{t-1} - \alpha(\lambda\mathbf{w}_{t-1} - \frac{1}{|A_t|}\sum_{(\mathbf{x},y)\in A_t^+} y\mathbf{x})\end{aligned} \tag{3}$$

where A_t^+ is the set of samples suffering non-zero losses, α is the learning rate, and $|A_t| = n$ is considered as the batch size. Furthermore, Shai et al. suggested the learning rate should decay with time, e.g. $\alpha = 1/t\lambda$, to guarantee convergence [5].

Pegasos has better speed, but it is likely to have lower accuracies on complex data sets due to its intrinsic linearity. In order to improve accuracies, many kernelized methods are proposed, such as KP, OGD, BOGD, NOGD, and FOGD. However, most of them have to update the kernel matrix when a new sample is coming, which brings expensive computational cost. Among of them, FOGD uses Fourier features to avoid constructing the kernel matrix. However, its accuracy and speed are limited by the number of Fourier components. That is, the more components sampled, the higher accuracy it obtains and the slower it runs [13].

3 Sketch discriminatively regularized online gradient descent classification

In this section, we will present our SDROGD, which is a linear online method based on Pegasos. In order to keep fast computational speed and high classification accuracy simultaneously, SDROGD focuses on taking full advantage of discriminative information rather than the kernelization. Besides introducing the information into the loss function, SDROGD directly embeds it into the regularization term and thus constructs a new discriminative regularization term.

3.1 Model construction

Given a training set S, we first measure the intra-class compactness and inter-class separation with the help of the within-class scatter matrix $\mathbf{S}_w$ and between-class scatter matrix $\mathbf{S}_b$ respectively.

These two scatter matrices are defined as follows [14]:

$$\mathbf{S}_b = \sum_{c\in\{-1,1\}} \frac{N_c}{N}(\mathbf{u}_c - \mathbf{u})(\mathbf{u}_c - \mathbf{u})^T \tag{4}$$

$$\mathbf{S}_w = \frac{\mathbf{X}^T\mathbf{X}}{N} - \mathbf{u}\mathbf{u}^T - \mathbf{S}_b \tag{5}$$

where $\mathbf{u}_c \in \mathbb{R}^{d\times 1}$ is the mean vector of class c, and N_c is the corresponding sample number. Moreover, $\mathbf{u} \in \mathbb{R}^{d\times 1}$ is the mean vector of all the samples, and N is the whole sample number. $\mathbf{X} \in \mathbb{R}^{N\times d}$ is the training matrix.

Then we define the discriminative information matrix $\mathbf{R}$ as:

$$\mathbf{R} = \eta\mathbf{S}_w - (1-\eta)\mathbf{S}_b \tag{6}$$

where η is a regularization parameter which controls the relative importance between $\mathbf{S}_w$ and $\mathbf{S}_b$, $0 \le \eta \le 1$.

Inspired by [15], we directly embed $\mathbf{R}$ into the regularization term. As a result, the objective function of SDROGD can be formulated as:

$$f(\mathbf{w}; S) = \frac{1}{|S|}\sum_{(\mathbf{x},y)\in S} l(\mathbf{w}; (\mathbf{x}, y)) + \frac{\lambda}{2}\mathbf{w}^T\mathbf{w} + \frac{1}{2}\mathbf{w}^T\mathbf{R}\mathbf{w} \tag{7}$$

where $\mathbf{w} \in \mathbb{R}^{d\times 1}$.

$\mathbf{w}^T\mathbf{R}\mathbf{w}$ is the new discriminative regularization term. Different from the Tikhonov smoothness regularization term, it directly starts from the essence of classification. More intuitively, $\mathbf{w}^T\mathbf{S}_b\mathbf{w}$ reflects the distance between the mean vector of each class and the global mean vector. The larger the value, the greater the distance between the different classes. Then $\mathbf{w}^T\mathbf{S}_w\mathbf{w}$ reflects the degree of separation of the samples within each class. The smaller the value, samples stay nearer in each class. Therefore, by minimizing the objective function (7), we could make the samples in the same class as near as possible while the samples of different classes as far as possible in the output space.

Just like Pegasos, SGD is applied to optimize the objective function $f(\mathbf{w}; S)$ in an online manner. The updating of $\mathbf{w}$ is:

$$\begin{aligned}\mathbf{w}_t &= \mathbf{w}_{t-1} - \alpha\nabla f(\mathbf{w}; A_t)\\ &= \mathbf{w}_{t-1} - \alpha(\lambda\mathbf{w}_{t-1} + \mathbf{R}\mathbf{w}_{t-1} - \frac{1}{|A_t|}\sum_{(\mathbf{x},y)\in A_t^+} y\mathbf{x})\end{aligned} \tag{8}$$

Now, it is worth to point out that there are two issues that need to be addressed. On the one hand, we have to update $\mathbf{S}_w$, $\mathbf{S}_b$, and $\mathbf{R}$ when new samples come due to the online setting. On the other hand, we have to cut down the cost of updating $\mathbf{w}$. Noticed that $\mathbf{S}_w$, $\mathbf{S}_b$, and $\mathbf{R}$ are all $d \times d$, if d is large, it will result in unaffordable storage and computational cost especially when updating $\mathbf{w}$ with $\mathbf{R}$ directly.

Actually, there are many methods to update $\mathbf{S}_w$, $\mathbf{S}_b$ in an online manner. For example, Pang et al. proposed a method to update $\mathbf{S}_w$ and $\mathbf{S}_b$ incrementally [16], but it has to preserve these two $d \times d$ matrices in memory. Ye et al. proposed such an incremental method based on QR decomposition [17], but there are still some $d \times d$ matrices that appear in memory. Both of them inevitably cause expensive cost on high-dimensional datasets. Another more reasonable choice is to address this problem by the sketch technique [18]. Combined with a simple optimization trick, we can solve the two above issues simultaneously. In the next two subsections, we will discuss the sketch technique firstly and then elaborate on how to cut down the computational cost.

3.2 Sketch technique In SDROGD

Sketch technique also works iteratively and obtains a mini-batch of samples at each iteration, which just coincides with the SGD framework. It maintains the main variations of all passed samples in a low-rank matrix $\mathbf{B} \in \mathbb{R}^{2l\times d}$, called as "sketch matrix", where l is a hyperparameter, $l \ll d$ and $\mathbf{B}$ is initialized by all zero. At iteration t, each new coming sample $\mathbf{x}_i$ will replace one zero row in $\mathbf{B}$ from top to bottom. Once $\mathbf{B}$ has no zero row, then the singular value decomposition (SVD) is performed on $\mathbf{B}$: $\mathbf{B} = \mathbf{U}\Sigma\mathbf{V}$, where $\Sigma \in \mathbb{R}^{2l\times 2l}$ is a diagonal matrix with all the singular values of $\mathbf{B}$ in decreasing order at its main diagonal, $\mathbf{V} \in \mathbb{R}^{2l\times d}$ and ith row of $\mathbf{V}$ is the corresponding singular vector of ith singular value in Σ. After that, we set:

$$\begin{aligned}\hat{\Sigma} &= \sqrt{\max(\Sigma^2 - \xi^2\mathbf{I}, \mathbf{O})}\\ \mathbf{B} &= \hat{\Sigma}\mathbf{V}\end{aligned} \tag{9}$$

where $\mathbf{O} \in \mathbb{R}^{2l\times 2l}$ is a zero matrix, $\mathbf{I} \in \mathbb{R}^{2l\times 2l}$ is an identity matrix, and ξ is the $(l+1)$th largest singular value. In this way, all the singular values are rescaled and the bottom half of $\mathbf{B}$ is reset to zero.

As the samples pass, the mean vectors $\mathbf{u}$ and $\mathbf{u}_c$ are also updated. Then one iteration ends up with returning $\mathbf{S}_b$ and the approx $\mathbf{S}_w$. $\mathbf{S}_b$ is still calculated as (4) while the approximation of $\mathbf{S}_w$ is calculated as follows:

$$\hat{\mathbf{S}}_w = \frac{\mathbf{C}^T\mathbf{C}}{N} - \mathbf{u}\mathbf{u}^T - \mathbf{S}_b \tag{10}$$

where $\mathbf{C} \in \mathbb{R}^{l\times d}$ is the upper half of $\mathbf{B}$. Finally, it waits for the next mini batch to start a new iteration.

It is worth to point out although $\hat{\mathbf{S}}_w$ is an approximation of the actual within-class scatter matrix $\mathbf{S}_w$, they are close enough. More detailed analysis is given in Section 4. Then the calculation of discriminative information matrix is changed to:

$$\hat{\mathbf{R}} = \eta\hat{\mathbf{S}}_w - (1-\eta)\mathbf{S}_b \tag{11}$$

So far, $\hat{\mathbf{R}}$ effectively involves the discriminative information of all the historical samples, which can be fully used in classification.

Consequently, the objective function of SDROGD is reformulated to:

$$\hat{f}(\mathbf{w}; S) = \frac{1}{|S|}\sum_{(\mathbf{x},y)\in S} l(\mathbf{w}; (\mathbf{x}, y)) + \frac{\lambda}{2}\mathbf{w}^T\mathbf{w} + \frac{1}{2}\mathbf{w}^T\hat{\mathbf{R}}\mathbf{w} \tag{12}$$

The corresponding updating for $\mathbf{w}$ is:

$$\begin{aligned}\mathbf{w}_t &= \mathbf{w}_{t-1} - \alpha\nabla\hat{f}(\mathbf{w}; A_t)\\ &= \mathbf{w}_{t-1} - \alpha(\lambda\mathbf{w}_{t-1} + \hat{\mathbf{R}}\mathbf{w}_{t-1} - \frac{1}{|A_t|}\sum_{(\mathbf{x},y)\in A_t^+} y\mathbf{x})\end{aligned} \tag{13}$$

3.3 Cutting down the cost

As mentioned above, $\hat{\mathbf{S}}_w$, $\mathbf{S}_b$, and $\hat{\mathbf{R}}$ are all $d \times d$ which still need to be preserved in memory. If d is large, it may cause memory overflow and serious time-consuming. Especially, when updating $\mathbf{w}$ according to (13), the product of $\hat{\mathbf{R}}$ and $\mathbf{w}_{t-1}$ takes $O(d^2)$ time, which may be unaffordable. Consequently, the only utilization of sketch technique still unable to avoid the high computational cost. Next, we will elaborate on how to solve this problem. For conciseness, here we omit the subscript of $\mathbf{w}_{t-1}$.

Firstly, we define:

$$\mathbf{v}_c = \sqrt{\frac{N_c}{N}}(\mathbf{u}_c - \mathbf{u}) \tag{14}$$

which is a $d \times 1$ vector. $c \in \{-1, 1\}$ denotes the class labels of two classes. Therefore, the between-class scatter matrix $\mathbf{S}_b$ can be rewritten as:

$$\mathbf{S}_b = [\mathbf{v}_{-1}, \mathbf{v}_1][\mathbf{v}_{-1}, \mathbf{v}_1]^T \tag{15}$$

With the help of (10) and (15), the product of $\hat{\mathbf{R}}$ and $\mathbf{w}$ can be calculated as follows:

$$\begin{aligned}\mathbf{z} &= \hat{\mathbf{R}}\mathbf{w} = [\eta\hat{\mathbf{S}}_w - (1-\eta)\mathbf{S}_b]\mathbf{w}\\ &= \eta(\frac{\mathbf{C}^T\mathbf{C}}{N} - \mathbf{u}\mathbf{u}^T - \mathbf{S}_b)\mathbf{w} - (1-\eta)\mathbf{S}_b\mathbf{w}\\ &= \eta(\frac{\mathbf{C}^T\mathbf{C}}{N} - \mathbf{u}\mathbf{u}^T)\mathbf{w} - \mathbf{S}_b\mathbf{w}\\ &= \eta\frac{\mathbf{C}^T\mathbf{C}}{N}\mathbf{w} - \mathbf{u}\mathbf{u}^T\mathbf{w} - [\mathbf{v}_{-1}, \mathbf{v}_1][\mathbf{v}_{-1}, \mathbf{v}_1]^T\mathbf{w}\end{aligned} \tag{16}$$

Reviewing that $\mathbf{C} \in \mathbb{R}^{l\times d}$, it will take $O(ld^2)$ to compute $\mathbf{C}^T\mathbf{C}$ and return a $d \times d$ result. The cost of this operation is expensive. The similar situation also exists when computing $\mathbf{u}\mathbf{u}^T$ and $\mathbf{S}_b$.

In fact, this problem can be easily solved by reordering the computation sequence. Considering that the product of matrix satisfies the associative law, we calculate $\mathbf{z}$ using the following order:

$$\mathbf{z} = \eta\frac{\mathbf{C}^T(\mathbf{C}\mathbf{w})}{N} - \mathbf{u}(\mathbf{u}^T\mathbf{w}) - [\mathbf{v}_{-1}, \mathbf{v}_1]\left([\mathbf{v}_{-1}, \mathbf{v}_1]^T\mathbf{w}\right) \tag{17}$$

In this way, it is unnecessary to preserve $\hat{\mathbf{S}}_w$ and $\mathbf{S}_b$ explicitly any more and thus no $d \times d$ matrix appears in memory. In addition, the time complexity is cut down to $O(ld)$. The pseudo code of SDROGD at iteration t is shown in algorithm 1.

Algorithm 1 Outline of SDROGD in iteration t.

Input: A_t: a mini-batch of samples
Input: $\mathbf{w}_{t-1}$: the obtained model parameters at iteration $t-1$.
Output: $\mathbf{w}_t$: updated model parameters.

for each $(\mathbf{x}, y) \in A_t$ **do**
  replace one zero row in $\mathbf{B}$ using $\mathbf{x}$
  if $\mathbf{B}$ has no zero row **then**
    $\mathbf{U}, \Sigma, \mathbf{V} \leftarrow \text{SVD}(\mathbf{B})$
    $\xi \leftarrow$ the $(l+1)$-th largest singular value in Σ
    $\hat{\Sigma} \leftarrow \sqrt{\max(\Sigma^2 - \xi^2\mathbf{I}, \mathbf{O})}$
    $\mathbf{B} \leftarrow \hat{\Sigma}\mathbf{V}$
  end if
  $\mathbf{u}_y \leftarrow N_y\mathbf{u}_y/(N_y+1)$
  $\mathbf{u} \leftarrow N\mathbf{u}/(N+1)$
  $N_y \leftarrow N_y + 1$
  $N \leftarrow N + 1$
end for
$\mathbf{C} \leftarrow$ the upper half of $\mathbf{B}$
for $c \in \{-1, 1\}$ **do**
  calculate $\mathbf{v}_c$ according to (14)
end for
calculate $\mathbf{z} = \hat{\mathbf{R}}\mathbf{w}_{t-1}$ according to (17)
update $\mathbf{w}_t$ in (13) with the help of $\mathbf{z}$
return $\mathbf{w}_t$

4 Theoretical analysis

4.1 The approximative property

In this subsection, we will show $\hat{f}(\mathbf{w}; S)$ is a good approximate of $f(\mathbf{w}; S)$. Firstly, from [19], we have the following lemma:

Lemma 1 *Given the whole training samples* $\mathbf{X} \in \mathbb{R}^{N\times d}$, *the upper half of sketch matrix* $\mathbf{C} \in \mathbb{R}^{l\times d}$, *then we have:* $\mathbf{X}^T\mathbf{X} \succeq \mathbf{C}^T\mathbf{C}$ *and* $\|\mathbf{X}^T\mathbf{X} - \mathbf{C}^T\mathbf{C}\| \le 2\|\mathbf{X}\|^2/l$.

Based on Lemma 1, we could obtain the following theorem immediately:

Theorem 1 *Given* $\mathbf{S}_w$ *calculated as* (5) *and* $\hat{\mathbf{S}}_w$ *calculated as* (10), *then for* $\forall \mathbf{w} \in \mathbb{R}^d$,

$$\mathbf{w}^T\hat{\mathbf{S}}_w\mathbf{w} \le \mathbf{w}^T\mathbf{S}_w\mathbf{w} \le \mathbf{w}^T\hat{\mathbf{S}}_w\mathbf{w} + 2\|\mathbf{w}\|^2\frac{\|\mathbf{X}\|^2}{Nl}$$

Proof Firstly, according to (5) and (10), we have:

$$\mathbf{S}_w - \hat{\mathbf{S}}_w = \frac{\mathbf{X}^T\mathbf{X} - \mathbf{C}^T\mathbf{C}}{N} \tag{18}$$

Combined with lemma 1:

$$0 \le \|\mathbf{S}_w - \hat{\mathbf{S}}_w\| \le \frac{2\|\mathbf{X}\|^2}{Nl} \tag{19}$$

Therefore

$$0 \le \mathbf{w}^T(\mathbf{S}_w - \hat{\mathbf{S}}_w)\mathbf{w} \le 2\|\mathbf{w}\|^2\frac{\|\mathbf{X}\|^2}{Nl} \tag{20}$$

which equals to

$$\mathbf{w}^T\hat{\mathbf{S}}_w\mathbf{w} \le \mathbf{w}^T\mathbf{S}_w\mathbf{w} \le \mathbf{w}^T\hat{\mathbf{S}}_w\mathbf{w} + 2\|\mathbf{w}\|^2\frac{\|\mathbf{X}\|^2}{Nl} \tag{21}$$

□

Theorem 1 shows the relationship between the approximate within-class scatter matrix $\hat{\mathbf{S}}_w$ and the actual $\mathbf{S}_w$. Without loss of generality, we could suppose $\|\mathbf{w}\| = 1$. Then combined with (7) and (12), we could have the following relation directly:

$$\hat{f}(\mathbf{w}; S) \le f(\mathbf{w}; S) \le \hat{f}(\mathbf{w}; S) + c \tag{22}$$

where c is a constant value irrelevant to $\mathbf{w}$, $c \ge 0$.

Now, we can find that sketch technique actually does not lose too much information, that is, $\hat{f}(\mathbf{w}; S)$ is a good approximation of $f(\mathbf{w}; S)$. Therefore, optimizing $\hat{f}(\mathbf{w}; S)$ has similar effect of optimizing $f(\mathbf{w}; S)$ directly.

4.2 Convergence property

When $\hat{\mathbf{R}}$ is semi-positive definite or λ greater than the absolute value of the minimal eigenvalue of $\hat{\mathbf{R}}$ at each iteration, in other words, $\hat{f}(\mathbf{w}; A_t)$ is convex, the following lemma gives the regret bound of SDROGD [20].

Lemma 2 *Assume* $\hat{f}(\mathbf{w}_t; A_t)$, $t = 1, 2, \ldots, T$ *be a sequence of convex functions, where* $\|\nabla\hat{f}(\mathbf{w}_t; A_t)\| \le G$, *and* $\|\mathbf{w}_t\|^2 \le D$. *Let* $\mathbf{w}^*$ *be the optimal solution of* $\hat{f}(\mathbf{w}; S)$, *then we have:*

$$\sum_{t=1}^{T}\hat{f}(\mathbf{w}_t; A_t) - \sum_{t=1}^{T}\hat{f}(\mathbf{w}^*; A_t) \le GD\sqrt{T} \tag{23}$$

For completeness, here we give the proof of Lemma 2.

Proof Firstly, we have the following equation according to (13)

$$\begin{aligned}&\|\mathbf{w}_{t+1} - \mathbf{w}^*\|^2 = \|\mathbf{w}_t - \alpha\nabla\hat{f}(\mathbf{w}; A_t) - \mathbf{w}^*\|^2\\ &= \|\mathbf{w}_t - \mathbf{w}^*\|^2 + \alpha^2\|\nabla\hat{f}(\mathbf{w}; A_t)\|^2 - 2\alpha\nabla\hat{f}(\mathbf{w}; A_t)\cdot\\ &\quad(\mathbf{w}_t - \mathbf{w}^*)\end{aligned} \tag{24}$$

Since $\hat{f}(\mathbf{w}_t; A_t)$ is a convex function, we have

$$\hat{f}(\mathbf{w}_t; A_t) - \hat{f}(\mathbf{w}^*; A_t) \le \nabla\hat{f}(\mathbf{w}_t; A_t)\cdot(\mathbf{w}_t - \mathbf{w}^*) \tag{25}$$

Rearrange (24) and substitute it into (25), we get

$$\hat{f}(\mathbf{w}_t; A_t) - \hat{f}(\mathbf{w}^*; A_t) \leq \frac{1}{2\alpha}(\|\mathbf{w}_t - \mathbf{w}^*\|^2 - \|\mathbf{w}_{t+1} - \mathbf{w}^*\|^2) + \frac{\alpha}{2}\|\nabla \hat{f}(\mathbf{w}; A_t)\|^2 \tag{26}$$

Supposing that α is a constant and summing up (26) over t from 1 to T, we get

$$\begin{aligned} &\sum_{t=1}^{T} \hat{f}(\mathbf{w}_t; A_t) - \sum_{t=1}^{T} \hat{f}(\mathbf{w}^*; A_t) \\ &\leq \frac{1}{2\alpha}(\|\mathbf{w}_1 - \mathbf{w}^*\|^2 - \|\mathbf{w}_T - \mathbf{w}^*\|^2) + \frac{\alpha T}{2}\|\nabla \hat{f}(\mathbf{w}; A_t)\|^2 \\ &\leq \frac{1}{2\alpha} D + \frac{\alpha T}{2} G \end{aligned} \tag{27}$$

Let $\alpha = \frac{D}{G\sqrt{T}}$, we will get the conclusion of Lemma 2. □

Otherwise, we could consider the objective $\hat{f}(\mathbf{w}; S)$ as a general non-convex function. In this case, the following lemma gives the convergence rate of SDROGD to an ϵ-accuracy stationary point (i.e. $\|\nabla \hat{f}(\mathbf{w}_T; S)\|^2 \leq \epsilon$) finally [21].

Lemma 3 *Suppose that (1) $\hat{f}(\mathbf{w}; S)$ is a Lipschitz-continuous gradients function, that is $|\hat{f}(\mathbf{w}_i; S) - \hat{f}(\mathbf{w}_j; S) - \nabla \hat{f}(\mathbf{w}_j; S) \cdot (\mathbf{w}_i - \mathbf{w}_j)| \leq \frac{L}{2}\|\mathbf{w}_i - \mathbf{w}_j\|^2, \forall i, j \in \{1, 2, \ldots, T\}$, $L > 0$ is a constant; (2) $\mathbb{E}[\|\nabla \hat{f}(\mathbf{w}; A_t)\|] = \nabla \hat{f}(\mathbf{w}; S)$; (3) $\mathbb{E}[\|\nabla \hat{f}(\mathbf{w}; A_t) - \nabla \hat{f}(\mathbf{w}; S)\|^2] \leq \sigma^2$;(4) the learning rate α is a constant less than $2/L$. Then the iterations T of SGD satisfies:*

$$\mathbb{E}[\|\nabla \hat{f}(\mathbf{w}_r; S)\|^2] \leq \sqrt{\frac{mL}{T}}\sigma \tag{28}$$

where $m = 2(\hat{f}(\mathbf{w}_1; S) - \hat{f}(\mathbf{w}^*; S))$, and r is chosen randomly from $1, 2, \ldots, T$.

Let $\epsilon = 1/\sqrt{T}$, then the complexity of SGD to obtain an ϵ-accuracy stationary point is $O(1/\epsilon^2)$. For completeness, we still give the proof of Lemma 3.

Proof Since $\hat{f}(\mathbf{w}; S)$ is Lipschitz-continuous gradients, we have

$$\hat{f}(\mathbf{w}_{t+1}; S) \leq \hat{f}(\mathbf{w}_t; S) + \nabla \hat{f}(\mathbf{w}_t; S) \cdot (\mathbf{w}_{t+1} - \mathbf{w}_t) + \frac{L}{2}\|\mathbf{w}_{t+1} - \mathbf{w}_t\|^2 \tag{29}$$

Now, let $\delta_t = \nabla \hat{f}(\mathbf{w}_t; A_t) - \nabla \hat{f}(\mathbf{w}_t; S)$, then combine with the updating (13), we have

$$\begin{aligned} &\hat{f}(\mathbf{w}_{t+1}; S) \leq \hat{f}(\mathbf{w}_t; S) + \nabla \hat{f}(\mathbf{w}_t; S) \cdot (\mathbf{w}_{t+1} - \mathbf{w}_t) + \frac{L}{2}\|\mathbf{w}_{t+1} - \mathbf{w}_t\|^2 \\ &= \hat{f}(\mathbf{w}_t; S) - \alpha \nabla \hat{f}(\mathbf{w}_t; S) \cdot \nabla \hat{f}(\mathbf{w}_t; A_t) + \frac{L}{2}\alpha^2 \|\nabla \hat{f}(\mathbf{w}_t; A_t)\|^2 \\ &= \hat{f}(\mathbf{w}_t; S) - (\alpha - \frac{L\alpha^2}{2})\|\nabla \hat{f}(\mathbf{w}_t; S)\|^2 - (\alpha - L\alpha^2)\nabla \hat{f}(\mathbf{w}_t; S)\delta_t \\ &\quad + \frac{L}{2}\alpha^2 \|\delta_t\|^2 \end{aligned} \tag{30}$$

Summing up the above inequality and re-arranging the terms, we obtain

$$\begin{aligned} &\sum_{t=1}^{T}(\alpha - \frac{L\alpha^2}{2})\|\nabla \hat{f}(\mathbf{w}_t; S)\|^2 \\ &\leq \hat{f}(\mathbf{w}_1; S) - \hat{f}(\mathbf{w}_{T+1}; S) - \sum_{t=1}^{T}(\alpha - L\alpha^2)\nabla \hat{f}(\mathbf{w}_t; S) \cdot \delta_t \\ &\quad + \frac{L}{2}\sum_{t=1}^{T}\alpha^2 \|\delta_t\|^2 \\ &\leq \hat{f}(\mathbf{w}_1; S) - \hat{f}(\mathbf{w}^*; S) - \sum_{t=1}^{T}(\alpha - L\alpha^2)\nabla \hat{f}(\mathbf{w}_t; S) \cdot \delta_t \\ &\quad + \frac{L}{2}\sum_{t=1}^{T}\alpha^2 \|\delta_t\|^2 \end{aligned} \tag{31}$$

Taking expectations on both sides of (31) and noting that $\mathbb{E}[\|\delta_t\|^2] \leq \sigma^2$ and $\mathbb{E}[\nabla \hat{f}(\mathbf{w}_t; S) \cdot \delta_t] = 0$, we obtain

$$\sum_{t=1}^{T}(\alpha - \frac{L\alpha^2}{2})\mathbb{E}[\|\nabla \hat{f}(\mathbf{w}_t; S)\|^2] \leq \hat{f}(\mathbf{w}_1; S) - \hat{f}(\mathbf{w}^*; S) + \frac{L\sigma^2 T\alpha^2}{2} \tag{32}$$

Now, supposing that r is chosen randomly from $1, 2, \ldots, T$, we have $T\mathbb{E}[\|\nabla \hat{f}(\mathbf{w}_r; S)\|^2] = \sum_{t=1}^{T}\mathbb{E}[\|\nabla \hat{f}(\mathbf{w}_t; S)\|^2]$. Therefore, (32) could be rewritten as

$$\begin{aligned} &\mathbb{E}[\|\nabla \hat{f}(\mathbf{w}_r; S)\|^2] \\ &\leq \frac{1}{T(2\alpha - L\alpha^2)}[2(\hat{f}(\mathbf{w}_1; S) - \hat{f}(\mathbf{w}^*; S)) + L\sigma^2 T\alpha^2] \\ &\leq \frac{1}{2\alpha T}[2(\hat{f}(\mathbf{w}_1; S) - \hat{f}(\mathbf{w}^*; S)) + L\sigma^2 T\alpha^2] \end{aligned} \tag{33}$$

Now, let $m = 2(\hat{f}(\mathbf{w}_1; S) - \hat{f}(\mathbf{w}^*; S))$ and $\alpha = \sqrt{\frac{m}{TL\sigma^2}}$, we have

$$\mathbb{E}[\|\nabla \hat{f}(\mathbf{w}_r; S)\|^2] \leq \sqrt{\frac{mL}{T}}\sigma \tag{34}$$

□

Table 1 The comparison of time/space complexity of related methods

	Pegasos [5]	KP [6]	BOGD [9]	NOGD [10]	FOGD [10]	BOLSSVM [12]	SDROGD
Time	$O(Nd)$	$O(N^2d)$	$O(NBd)$	$O(NB(d+B^2))$	$O(NBd)$	$O(NB(d+B))$	$O(Ndl)$
Space	$O(d)$	$O(Nd)$	$O(Bd)$	$O(Bd)$	$O(Bd)$	$O(Bd)$	$O(dl)$

N means the total number of samples. d means the number of features. B is the budget of corresponding methods. $l \ll d$ is a hyperparameter in SDROGD

4.3 Complexity analysis

Following the discussion in Section 3, when a mini batch of samples come and make **B** full, the method will perform the sketch process. The costly operation is SVD on **B**, which takes $O(l^2d)$ time.

Then the method will calculate **z** according to (17) and update **w**. It only contains dot product operations and takes $O(ld)$ time.

In summary, during the training phase, **B** is full N/l times in total and takes $O(l^2d)$ each time. Moreover, there are N/n mini batches and each mini batch takes $O(ld)$ to update **w**. As a result, the computational complexity of SDROGD is $O(Ndl)$ in total.

As for space complexity, the largest variable that needs to be stored is the sketch matrix **B**, so the space complexity is $O(dl)$.

For clarity, we compare SDROGD with some related online classification methods in time and space complexity, which is shown in Table 1.

5 Experiments

5.1 Datasets and preprocessing

We evaluate our SDROGD on ten datasets, including one toy dataset and nine public datasets, that is a9a [22], cifar-10 [23], cod-rna [24], german [25], gisette [26], ijcnn1 [27], madelon [26], mnist [28], and real-sim [25]. Since cifar-10 and mnist datasets are multi-class, we randomly choose two classes for training. Concretely, we use dog vs. horse for cifar-10 dataset, and 0 vs. 1 for mnist dataset.

Cifar-10[1] and mnist[2] datasets can be downloaded at their official websites respectively. All the rest of public datasets can be downloaded at LIBSVM[3] or UCI repository[4]. The toy dataset is generated by the sklearn [29] API: *make_classification*. It includes 8000 samples with 200 dimensions, which could be considered as two 200-dimension Gaussian distribution with some noise. To be more intuitive, we random select 400 samples from toy dataset and visualize it after principal components analysis (PCA), which is shown in Fig. 1.

We perform standardization on each dataset except a9a and toy datasets. Standardization means removing the mean and scaling to a unit variance for each feature. We transform labels to $\{-1, 1\}$ to satisfy the models' input requirement. Table 2 shows the details of all the datasets we used.

5.2 Experimental setup and results

We compare our method with other six methods: Pegasos [5], KP [6], BOGD [9], NOGD [10], FOGD [10], and BOLSSVM [12]. KP, BOGD, NOGD, and BOLSSVM have both linear and Gaussian kernel versions. Pegasos only has the linear version and FOGD only has the Gaussian kernel version.

We set hyperparameters on datasets as follows. The learning rate of Pegasos is set to $1/t\lambda$ following [5], while the learning rate of the rest methods are set to $1/t$. The batch size n of Pegasos and SDROGD is set to 60 while the others are 1. The budget g in BOGD, NOGD and BOLSSVM is set to 100. The number of Fourier components f in FOGD is also set to 100. The band width σ of radial basis

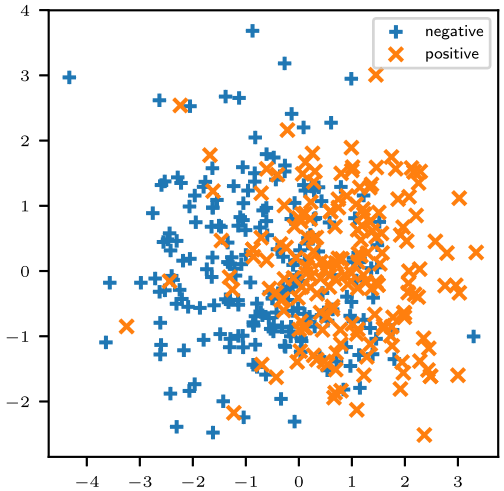

Fig. 1 400 random selected samples of toy dataset after PCA. The original feature space is 400-dimension

[1] https://www.cs.toronto.edu/~kriz/cifar.html

[2] http://yann.lecun.com/exdb/mnist/

[3] https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/

[4] http://archive.ics.uci.edu/ml/datasets.html

Table 2 The numbers of pos(itive) samples, neg(ative) samples and dim(ensions) of each dataset

	#pos	#neg	#dim	Standardized
a9a	11687	37155	123	NO
cifar-10	1977	2025	3072	YES
cod-rna	19845	39690	8	YES
german	300	700	24	YES
gisette	3500	3500	5000	YES
ijcnn1	13565	128126	22	YES
madelon	1300	1300	500	YES
mnist	7877	6903	784	YES
real-sim	22238	50071	20958	YES
toy	4003	3997	200	NO

function(rbf) used in the kernelized methods is searched from $\{2^{-3}, 2^{-2}, \ldots, 2^3\}$. The coefficient of the regularization term λ in Pegasos, BOLSSVM, and SDROGD is searched from $\{0, 2^{-3}, 2^{-2}, \ldots, 2^3\}$(0 is available only for SDROGD). The coefficient γ in BOGD is searched from $\{1, 10, 100\}$. The rank K of the approximate kernel matrix in NOGD is searched from $\{0.1g, 0.2g, \ldots, 0.5g\}$. The row number l of sketch matrix in SDROGD is searched from $\{0.01d, 0.1d, 0.2d, \ldots, 0.5d\}$, η is searched from $\{0, 0.1, \ldots, 1.0\}$.

Because the real-sim dataset is too large, it is too time-consuming to perform grid search on it. Instead, we adopt a random grid search from the same space of hyperparameters. We randomly select ten different

Table 3 Comparison of average accuracy(%) on test sets

	a9a	cifar-10	cod-rna	german	gisette
Pegasos	75.90 ± 0.38•	67.57 ± 1.50	66.59 ± 0.21•	67.10 ± 2.16•	88.33 ± 2.80•
KP(ln)	78.32 ± 4.06•	64.84 ± 3.35	85.65 ± 3.45•	55.50 ± 2.21•	94.96 ± 1.33
KP(rbf)	76.30 ± 9.40	**68.94 ± 1.34**	92.97 ± 0.54	59.20 ± 18.68	90.90 ± 0.56•
BOGD(ln)	75.82 ± 0.47•	59.38 ± 4.29•	70.74 ± 5.30•	59.10 ± 9.63•	88.90 ± 2.47•
BOGD(rbf)	75.90 ± 0.38•	60.35 ± 1.71•	79.48 ± 1.63•	67.70 ± 1.99•	80.53 ± 5.75•
NOGD(ln)	81.49 ± 0.51•	63.22 ± 3.01	87.30 ± 1.96•	62.70 ± 5.62•	93.66 ± 0.96•
NOGD(rbf)	76.34 ± 0.72•	60.42 ± 1.25•	79.17 ± 2.14•	70.10 ± 4.55	79.64 ± 3.59•
FOGD	82.47 ± 0.90	49.99 ± 1.34•	**94.26 ± 0.27**○	64.50 ± 7.51•	50.17 ± 0.57•
BOLSSVM(ln)	77.14 ± 2.23•	50.24 ± 1.83•	90.17 ± 1.16•	68.70 ± 6.82	48.24 ± 1.82•
BOLSSVM(rbf)	77.15 ± 3.11•	50.14 ± 1.84•	90.67 ± 0.96•	66.90 ± 2.16•	50.39 ± 1.06•
SDROGD	**83.49 ± 0.30**	65.77 ± 2.85	93.03 ± 0.55	**74.90 ± 1.34**	**96.40 ± 0.63**
	ijcnn1	madelon	mnist	real-sim	toy
Pegasos	90.40 ± 0.07	49.08 ± 0.92•	92.02 ± 0.79•	69.61 ± 0.33•	49.64 ± 0.81•
KP(ln)	56.43 ± 1.36•	52.65 ± 1.17•	99.63 ± 0.11•	NA	64.99 ± 7.31•
KP(rbf)	**98.24 ± 0.18**○	55.92 ± 1.04	99.63 ± 0.14•	NA	64.46 ± 5.41•
BOGD(ln)	50.89 ± 2.44•	52.27 ± 0.52•	99.63 ± 0.07•	70.06 ± 2.20•	59.29 ± 2.16•
BOGD(rbf)	90.38 ± 0.16	51.08 ± 1.97•	94.91 ± 1.08•	39.86 ± 15.03•	54.00 ± 3.01•
NOGD(ln)	60.26 ± 3.87•	50.54 ± 2.00•	99.72 ± 0.09	69.18 ± 0.58•	60.56 ± 4.32•
NOGD(rbf)	90.46 ± 0.17	52.96 ± 2.80•	99.14 ± 0.28•	36.74 ± 15.39•	54.37 ± 3.05•
FOGD	95.46 ± 0.30○	50.92 ± 2.16•	87.12 ± 0.24•	50.29 ± 0.75•	53.79 ± 0.86•
BOLSSVM(ln)	89.29 ± 1.60	53.50 ± 3.28•	87.83 ± 19.48	59.73 ± 19.48•	52.62 ± 5.60•
BOLSSVM(rbf)	92.12 ± 0.85○	51.85 ± 3.52•	98.72 ± 0.27•	59.46 ± 19.00•	55.81 ± 3.70•
SDROGD	90.40 ± 0.13	**57.54 ± 1.72**	**99.81 ± 0.09**	**92.96 ± 0.28**	**78.46 ± 5.20**

"NA" means we don't evaluate. "ln" means method using linear kernel and "rbf" means method using Gaussian kernel. In addition, •/○ indicates whether SDROGD is statistically superior/inferior to the compared algorithm on each dataset (pairwise t-test at 0.05 significance level)

Table 4 Comparison of training time (seconds)

	a9a	cifar-10	cod-rna	german	gisette	ijcnn1	madelon	mnist	real-sim	toy
Pegasos	0.76	0.36	0.17	0.03	0.77	0.47	0.04	0.44	18.16	0.25
KP(ln)	146.30	24.99	32.18	0.39	31.35	813.21	2.61	1.67	NA	4.13
KP(rbf)	371.69	81.74	33.32	0.72	108.46	123.67	5.38	2.89	NA	17.12
BOGD(ln)	5.99	4.36	7.99	0.38	15.72	22.93	0.33	2.37	895.20	1.62
BOGD(rbf)	12.58	25.28	14.91	0.34	36.30	43.99	0.61	17.05	1731.65	3.48
NOGD(ln)	4.90	6.47	5.38	0.58	15.86	19.33	0.27	1.87	894.95	1.10
NOGD(rbf)	9.32	16.57	7.04	0.13	35.72	22.89	0.45	5.84	1778.34	1.67
FOGD	4.10	2.43	5.24	0.64	10.05	14.03	0.26	2.26	587.84	0.88
BOLSSVM(ln)	10.99	31.69	12.96	0.32	25.16	28.43	1.28	2.00	602.87	2.71
BOLSSVM(rbf)	12.99	20.83	8.64	0.27	98.28	20.56	1.63	4.78	1726.47	3.11
SDROGD	3.52	25.82	2.92	0.09	68.75	4.75	0.95	3.95	544.10	0.72

"NA" means we don't evaluate. "ln" means method using linear kernel and "rbf" means method using Gaussian kernel

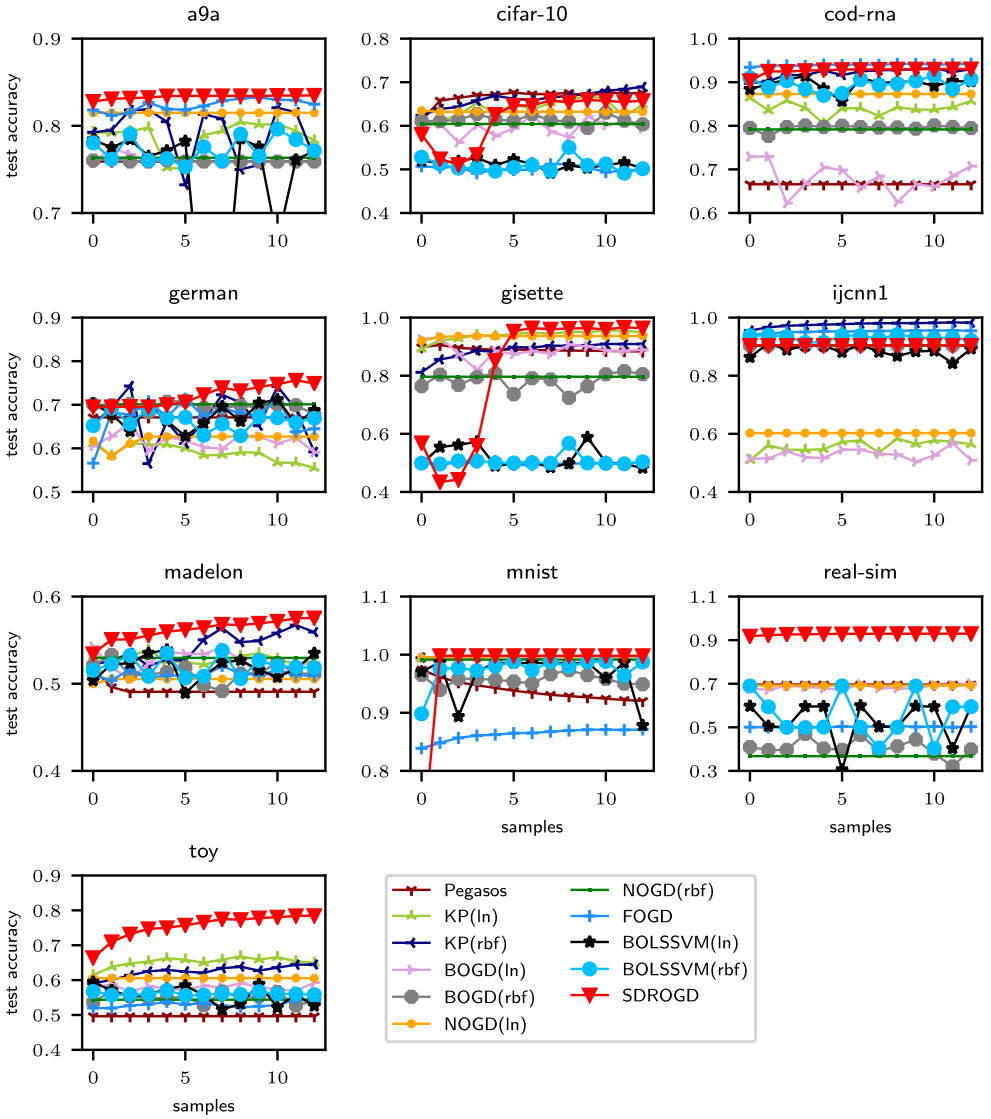

Fig. 2 The variation of test accuracy over iterations on each dataset

combinations of hyperparameters for each method and then choose the best. KP is not evaluated on the real-sim dataset, since its expensive cost.

For all datasets, we randomly select 80% samples as the training set and the rest 20% samples are considered as the test set. Then we select hyperparameters by 3-fold cross validation on the training set. This process repeats five epochs. We record the two property in each epoch: (1)the accuracy on test set; (2) the training time on the training set.

We report the average values in Tables 3 and 4. Besides, we record accuracy variations as iterations on the test set in Fig. 2. To show statistical significance, pairwise t-test at 0.05 significance level is conducted between the algorithms. Specifically, when SDROGD is significantly superior/inferior to the compared algorithm on any dataset, a marker •/◦ is shown. Otherwise, no marker is given [30]. In addition, we compare the second best method with our SDROGD on those datasets where SDROGD is the best according to the accuracy results. Then, we plot the ROC curves and give the values of AUC respectively. The results are shown in Fig. 3.

All the code is written in python. Our experiments run on a Linux server with 2.40GHz Xeon(R) CPU E5-2680.

From Tables 3 and 4, we can see that SDROGD has much better accuracies and good speed, especially on high-dimensional datasets, such as gisette, madelon, and real-sim datasets. The dimensions of these datasets are very high, which implies they are more likely tend to be linearly separable. SDROGD keeps the linearity of Pegasos and embeds discriminative information into the new regularization term, therefore, it is usually superior to the others in such high-dimensional datasets. Furthermore, SDROGD has even better accuracies than some kernelized methods, and it usually runs faster than the kernelized methods on average.

Figure 2 shows the variations of accuracies over iterations on each dataset. SDROGD often has better convergence. The accuracies of SDROGD on test sets increase steadily over time and often reach higher scores finally.

5.3 The effect of hyperparameters

From the results above, we can see that SDROGD outperforms the others, especially on the real-sim dataset. In this section, we will explore the influence of some

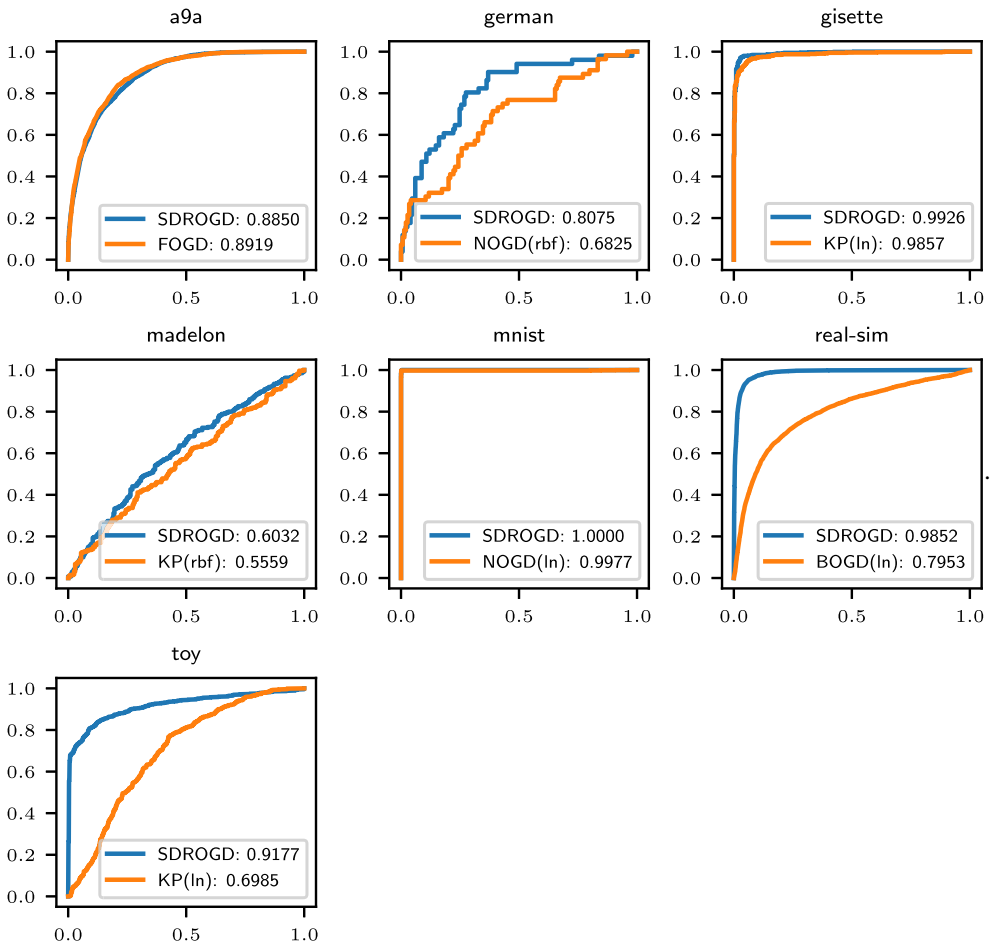

Fig. 3 The ROC curves of SDROGD and the second best method. The legend of each sub-figure indicates the name of method and the corresponding AUC

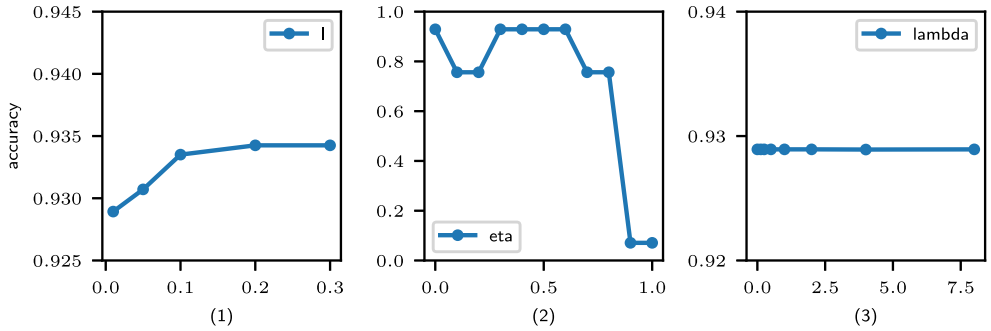

Fig. 4 The effect of some hyperparameters on real-sim dataset about accuracy. The best setting given by random search is $l = 0.01d$, $\eta = 0.3$, and $\lambda = 0$. We adjust these three hyperparameters respectively while fix the rests. (1) $l \in \{0.01, 0.05, 0.1, 0.2, 0.3\}$; (2) $\eta \in \{0.0, 0.1, \ldots, 1.0\}$; (3) $\lambda \in \{0, 2^{-3}, 2^{-2}, \ldots, 2^{3}\}$

hyperparameters on accuracy on the real-sim dataset. Based on the best hyperparameters combination given by random grid search, we adjust l, η, λ respectively while fixing the others and record the variations of accuracies.

We random select 66.7% samples for training and 33.3% for testing. After repeating 3 times, we compute the average accuracy score.

The results are shown in Fig. 4. We could see that SDROGD is insensitive to λ and l. In the course of the experiment, we also find that setting λ to 0 will produce good results in most cases. l controls how much information the sketch matrix maintains, so the larger l, the more accuracy in general, but a bigger l will brings heavier computational burden. One needs to weigh performance against accuracy when setting l.

From the second sub-figure of Fig. 4, it seems like η is a more important hyperparameter. Although SDROGD gives much low accuracy when η equals to 0.9 or 1.0, it performs well in most cases.

6 Extending to multi-class

Although in this paper we elaborate SDROGD in binary-class setting, it is possible to extend it to deal with multi-class problems. This extension just needs some small changes. Firstly, according to [31], we change the model parameter w from a $d \times 1$ vector to a $d \times C$ matrix, if there are C different classes. Secondly, the loss function should be changed to multi-class hinge loss [32] now. The discriminative information matrix $\mathbf{R}$ support multi-class inherently, and we could use it almost directly. Finally, we could get C classification hyperplanes after training only one time.

7 Conclusion

In this paper, we present an online linear classification method called SDROGD, which aims to deal with large-scale, including high-dimensional datasets. In order to obtain higher accuracy as well as keep a fast speed, we directly embed the discriminative information matrix into a new regularization term, which is constructed with between-class and within-class scatter matrices. Sketch technique is introduced to update this matrix in an online manner. By doing so, SDROGD can minimize the classification error and the intra-class compactness, and maximize the inter-class separation simultaneously. We also give theoretical analyses about the approximative property of sketch technique, convergence property, and computational complexity. Experimental results on many datasets compared with other state-of-the-art methods validate that SDROGD has better classification performance even than some kernelized methods. Besides, we also show that SDROGD has good stationarity as for some hyperparameters. Finally, we show that it is also possible for SDROGD to deal with multi-class problems.

Acknowledgments This work was supported by the National Key R&D Program of China (Grant No. 2017YFB1002801), the National Natural Science Foundations of China (Grant No. 61876091).

It is also supported by the Collaborative Innovation Center of Wireless Communications Technology.

References

1. Frank R (1958) The perceptron: a probabilistic model for information storage and organization in the brain. Psychol Rev 65(6):386
2. Shai S-S (2012) Online learning and online convex optimization. Found Trends® Mach Learn 4(2):107–194
3. Shi T, Zhu J (2017) Online bayesian passive-aggressive learning. J Mach Learn Res 18(1):1084–1122
4. Koby C, Ofer D, Joseph K, Shai S-S, Yoram S (2006) Online passive-aggressive algorithms. J Mach Learn Res 7:551–585
5. Shai S-S, Yoram S, Nathan S, Andrew C (2011) Pegasos: Primal estimated sub-gradient solver for svm. Math Program 127(1):3–30
6. Freund Y, Schapire RE (1999) Large margin classification using the perceptron algorithm. Mach Learn 37(3):277–296

7. Kivinen J, Smola AJ, Williamson RC (2002) Online learning with kernels. In: Advances in Neural Information Processing Systems, pp 785–792
8. Tu DN, Le T, Bui H, Phung DQ (2017) Large-scale online kernel learning with random feature reparameterization. In: Proceedings of the 26th International Joint Conference on Artificial Intelligence, pp 2543–2549
9. Wang Z, Koby C, Slobodan V (2012) Breaking the curse of kernelization Budgeted stochastic gradient descent for large-scale svm training. J Mach Learn Res 13(Oct):3103–3131
10. Lu J, Hoi Steven CH, Wang J, Zhao P, Liu Z-Y (2016) Large scale online kernel learning. J Mach Learn Res 17(1):1613–1655
11. Bo Y, Shao Q-M, Li P, Li W-B (2018) A study on regularized weighted least square support vector classifier. Pattern Recogn Lett 108:48–55
12. Jian L, Shen S, Li J, Liang X, Li Lei (2017) Budget online learning algorithm for least squares svm. IEEE Trans Neural Netw Learn Syst 28(9):2076–2087
13. Li Z, Ton J-F, Oglic D, Sejdinovic D (2018) Towards a unified analysis of random fourier features. arXiv:1806.09178
14. Kim T-K, Wong S-F, Bjorn S, Josef K, Roberto C (2007) Incremental linear discriminant analysis using sufficient spanning set approximations. In Computer Vision and Pattern Recognition. IEEE, pp 1–8
15. Xue H, Chen S, Yang Q (2009) Discriminatively regularized least-squares classification. Pattern Recogn 42(1):93–104
16. Pang S, Seiichi O, Nikola K (2005) Incremental linear discriminant analysis for classification of data streams. IEEE Trans Syst Man Cybern Part B Cybern 35(5):905
17. Ye J, Li Q, Xiong H, Haesun P, Ravi J, Kumar V (2005) Idr/qr: an incremental dimension reduction algorithm via qr decomposition. IEEE Trans Knowl Data Eng 17(9):1208–1222
18. Li W-H, Zhong Z, Zheng W-S (2017) One-pass person re-identification by sketch online discriminant analysis. arXiv:1711.03368
19. Edo L (2013) Simple and deterministic matrix sketching. In: Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. ACM, pp581–588
20. Bottou L, Curtis FE, Nocedal J (2018) Optimization methods for large-scale machine learning. SIAM Rev 60(2):223–311
21. Reddi SJ, Hefny A, Sra S, Poczos B, Smola A (2016) Stochastic variance reduction for nonconvex optimization. In: International conference on machine learning, pp 314–323
22. Kohavi R (1996) Scaling up the accuracy of naive-bayes classifiers: a decision-tree hybrid. In: Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining, vol 96, pp 202–207
23. Alex K, Geoffrey H (2009) Learning multiple layers of features from tiny images. Technical report, Citeseer
24. Sören S, Vojtech F (2010) Coffin: A computational framework for linear svms. In: Proceedings of the 27th International Conference on Machine Learning, pp 999–1006
25. Chang C-C, Lin C-J (2011) LIBSVM: A library for support vector machines. ACM Transactions on Intelligent Systems and Technology, pp 2:27:1–27:27. Software available at http://www.csie.ntu.edu.tw/cjlin/libsvm
26. Isabelle G, Steve G, Asa B-H, Gideon D (2005) Result analysis of the nips 2003 feature selection challenge. In: Advances in Neural Information Processing Systems, pp 545–552
27. Danil P (2001) Ijcnn 2001 neural network competition. Slide Present IJCNN 1:97
28. LeCun Y, Bottou L, Bengio Y, Haffner P (1998) Gradient-based learning applied to document recognition. Proc IEEE 86(11):2278–2324
29. Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, Blondel M, Prettenhofer P, Weiss R, Dubourg V, Vanderplas J, Passos A, Cournapeau D, Brucher M, Perrot M, Duchesnay E (2011) Scikit-learn: Machine learning in Python. J Mach Learn Res 12:2825–2830
30. Xu H-M, Xue H, Chen X-H, Wang Y-Y (2017) Solving indefinite kernel support vector machine with difference of convex functions programming. In Thirty-First AAAI Conference on Artificial Intelligence
31. Koby C, Yoram S (2001) On the algorithmic implementation of multiclass kernel-based vector machines. J Mach Learn Res 2(Dec):265–292
32. Kamiya R, Washizawa Y (2018) Discriminative sparse representation learning using multiclass hinge loss. In: 2018 Asia-pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC). IEEE, pp 955–958

Hui Xue received the B.Sc. degree in Mathematics from Nanjing Normal University in 2002. In 2005, she received the M.Sc. degree in Mathematics from Nanjing University of Aeronautics & Astronautics (NUAA). And she also received the Ph.D. degree in Computer Application Technology at NUAA in 2008. Since 2009, as an Associate Professor, she has been with the School of Computer Science and Engineering at Southeast University. Her research interests include pattern recognition and machine learning.

Zhen Ren is currently a postgraduate student majoring in Software Engineering in the School of Computer Science and Engineering at Southeast University. He received the bachelor's degree in Software Engineering from Nanjing University of Aeronautics & Astronautics (NUAA) in 2017. His research interests include pattern recognition and machine learning.