



Triply stochastic gradient method for large-scale nonlinear similar unlabeled classification

Wanli Shi^{1,2} · Bin Gu^{1,2,3} · Xiang Li⁴ · Cheng Deng⁵ · Heng Huang^{3,6}

Received: 23 November 2019 / Revised: 5 April 2021 / Accepted: 12 April 2021 /

Published online: 6 July 2021

© The Author(s), under exclusive licence to Springer Science+Business Media LLC, part of Springer Nature 2021

Abstract

Similar unlabeled (SU) classification is pervasive in many real-world applications, where only similar data pairs (two data points have the same label) and unlabeled data points are available to train a classifier. Recent work has identified a practical SU formulation and has derived the corresponding estimation error bound. It evaluated SU learning with linear classifiers on medium-sized datasets. However, in practice, we often need to learn nonlinear classifiers on large-scale datasets for superior predictive performance. How this could be done in an efficient manner is still an open problem for SU classification. In this paper, we propose a scalable kernel learning algorithm for SU classification using a triply stochastic optimization framework, called TSGSU. Specifically, in each iteration, our method randomly samples an instance from the similar pairs set, an instance from the unlabeled set, and their random features to calculate the stochastic functional gradient for the model update. Theoretically, we prove that our method can converge to a stationary point at the rate of $O(1/\sqrt{T})$ after T iterations. Experiments on various benchmark datasets and high-dimensional datasets not only demonstrate the scalability of TSGSU but also show the efficiency of TSGSU compared with existing SU learning algorithms while retaining similar generalization performance.

Keywords Weakly-supervised learning · SU classification · Kernel method · Large-scale optimization

1 Introduction

The supervised classification has achieved great success in many real-world applications, such as image recognition, speech recognition, and recommendation. It usually needs large amounts of labeled data to train a classifier. However, in many other practical applications, such as disaster resilience, medical diagnosis, and bioinformatics, massive labeled data cannot be collected easily since manually labeling the unlabeled data is time-consuming

Editor: Zhi-Hua Zhou.

✉ Bin Gu
bin.gu@mbzuai.ac.ae

Extended author information available on the last page of the article

Table 1 Existing SU classification algorithms

Algorithm	References	Loss function	Computational Com.	Space Com.
SUSL	Bao et al. (2018)	Squared loss	$O(n^3)$	$O(n^2)$
SUDH	Bao et al. (2018)	Double hinge loss	$O(n^3)$	$O(n^2)$
CUU	Lu et al. (2020)	All	$O(ndbT)$	$O(nb)$
TSGSU	Ours	All	$O(DT^2)$	$O(T)$

(D denotes the number of random features, n denotes the number of training samples, T denotes number of iterations and b denotes the batch size)

and laborious. To handle this problem, great efforts have been done in weakly-supervised classification, including semi-supervised learning (Chapelle et al. 2009; Sakai et al. 2017, 2018; Geng et al. 2019; Shi et al. 2019; Yu et al. 2019), positive unlabeled (PU) learning (du Plessis et al. 2014, 2015b, a) and one-class classification (Khan and Madden 2009; Schölkopf et al. 2001).

Recently, a new weakly-supervised classification problem called similar unlabeled (SU) classification (Bao et al. 2018) has been proposed. It contains two categories of datasets. One is the similar data pairs set, where ‘similar’ means that two data points belong to a certain (unknown) class, the other is the unlabeled dataset. The goal of SU classification is to utilize similar data pairs and the unlabeled set to learn a classifier. This problem naturally arises in the prediction of people’s sensitive matters such as religion, politics, and opinions on some racial issues. In these applications, people may hesitate to give explicit answers to these questions. However, it is easy to collect the answers that they have the same reply as someone else. Besides, massive unlabeled data can be easily collected from websites.

In many real-world learning applications, the datasets are not linearly separable, and directly using the linear model may ignore the non-linear structure of datasets which leads to a bad performance on prediction. To sufficiently capture the nonlinear data structures, the kernel method is one of the most common methods. Specifically, the kernel method maintains the kernel matrix to map the data into higher dimensional spaces such that the datasets become linearly separable. However, the kernel method often suffers from $O(n^2d)$ computational complexity and $O(n^2)$ space complexity to maintain the kernel matrix, where n denotes the total number of training sets and d denotes the data dimension. This makes it hardly scalable for SU learning methods proposed in Bao et al. (2018) when introduced the kernel method. Even worse, SUSL (Bao et al. 2018) needs to calculate the inverse of the kernel matrix when calculating the analytical solution, which has the computational complexity of $O(n^3)$. Recently, Lu et al. (2019) have proposed a gradient-based algorithm to solve the large-scale SU problem. It randomly samples two kinds of samples and then calculates the gradient of the objective function to update the model. However, it still needs the calculation of kernel function on the data batch, which makes the improvement of efficiency is limited. We summarize existing SU algorithms in Table 1. Comparing with the algorithms in Table 1, we can easily find that scaling up non-linear SU classification is still an open challenge.

To scale up kernel-based algorithms, a large amount of methods has been proposed, e.g., asynchronous parallel algorithms (Gu et al. 2016, 2018a), kernel approximation (Rahimi and Recht 2008; Smola and Schölkopf 2000). Recently, doubly stochastic gradient descent method (DSG) (Dai et al. 2014) has been successfully applied to many machine learning problems, such as semi-supervised learning (Shi et al. 2019; Geng

et al. 2019; Shi et al. 2020) and multiple kernels learning (Li et al. 2017). The standard DSG method randomly samples a data point and its random features in each iteration to calculate the functional gradient for model updating. However, the standard DSG framework and its theoretical analysis can not be applied to solve the SU classification problem. On the one hand, the SU problem contains two different data sources, *i.e.*, similar pairs set and unlabeled dataset, while the standard DSG focuses on optimizing the empirical risk on a single data source. On the other hand, the objective function of the SU learning is non-convex if the Consistent Correction Function (Lu et al. 2020) is added to avoid the negative risk. By using such a non-convex formulation, the previous analysis for the convex problem cannot be straightforwardly applied.

To address these challenging problems, our method is to add another stochastic layer of sampling into the standard DSG framework. Specifically, in each iteration, we randomly sample an instance from the similar pairs set and another instance from the unlabeled set to constitute a random data pair. Then the random features of this data pair can be generated on the fly by using a random number generator. With these three sources of randomness, *i.e.*, the similar sample, the unlabeled sample, and their random features, we can easily calculate the approximated stochastic functional gradient to update the model function. Since our proposed method contains three sources of randomness, we denote our method as Triply Stochastic Gradient descent for SU Classification (TSGSU). Theoretically, we give a new theoretical analysis based on the framework in Geng et al. (2019), Dai et al. (2014) and prove that our method can converge to the stationary point at the rate of $O(\frac{1}{\sqrt{T}})$ after T iterations. Our experiments

on various benchmark datasets and high-dimensional datasets not only demonstrate the scalability but also show the efficiency of TSGSU compared with existing learning algorithms while retaining similar generalization performance.

Contributions The main contributions of this paper are summarized as follows,

1. We propose an efficient algorithm to solve the large-scale kernel-based SU classification problem. Specifically, our method randomly samples a similar point, an unlabeled, and their random features to calculate the functional gradient and update the model. Compared with other methods, our method highly reduces computational complexity and space complexity.
2. In this paper, we established a new convergence analysis framework and prove that our TSGSU can converge to the stationary point at the rate $O(1/\sqrt{T})$ after T iterations for the non-convex SU problem. The standard DSG needs the convexity assumption of the objective function which is restrictive in real-world applications. On the contrary, our convergence rate is more general, even if it is slower than that of convex problems.

We organize the rest of our paper as follows. We begin by reviewing several related works in sect. 2. In sect. 3, we introduce the problem setting of SU classification and give a brief review of random Fourier features. In sect. 4, we propose our TSGSU algorithm. In sect. 5, we analyze the convergence rate of the proposed TSGSU. Experimental results on various datasets are discussed in sect. 6. Detailed proofs of the convergence rate are given in sect. 7. Finally, we make some conclusions in sect. 8.

2 Related works

In this section, we will briefly review several existing kernel approximation methods and compare SU classification with other weakly supervised learning problems.

2.1 Kernel approximation

Kernel approximation has been widely used to scale up kernel-based learning algorithms and existing methods can be decomposed into two categories. The first category is data-dependent methods, such as greedy basis selection techniques (Smola and Schölkopf 2000), incomplete Cholesky decomposition (Fine and Scheinberg 2001), Nyström method (Drineas and Mahoney 2005), which compute a low-rank approximation of the kernel matrix. However, this kind of method needs large amounts of samples to achieve a better generalization performance. Another category called data-independent methods, *e.g.*, random Fourier feature (RFF) (Rahimi and Recht 2008), orthogonal random feature (Yu et al. 2016) and quadrature based random feature (Munkhoeva et al. 2018), directly approximates the kernel function unbiasedly with some basis functions. Among the three kinds of random feature methods, RFF is the most commonly used and easiest to generate since orthogonal random feature and quadrature-based random feature need extra operations to calculate the orthogonal matrix. However, RFF needs to keep large amounts of random features in the memory to achieve a low approximation error. To further improve RFF, Dai et al. (2014) proposed DSG algorithm. It uses *pseudo-random number generators* to generate the random features on the fly, which highly reduces the memory requirement of RFF. Due to its superior performance, DSG has been successfully applied to scale up kernel-based algorithms in many applications, such as (Gu et al. 2018b; Li et al. 2017; Rahimi and Recht 2009; Le et al. 2013; Shi et al. 2019; Geng et al. 2019). The theoretical analysis of Dai et al. (2014), Gu et al. (2018b), Li et al. (2017), Shi et al. (2019) are all based on the assumption that the objective functions of these problems are convex. However, in SU learning, we consider a more general formulation, which is non-convex. This makes the standard theoretical analysis framework of DSG cannot be directly used in SU learning.

2.2 Weakly supervised learning

In order to train a binary classifier with high predictive performance, a vast amount of labeled data is needed. However, in many real-world applications, manually labeling the unlabeled instances is time-consuming. To address this issue, weakly supervised learning problems have been widely explored. Compared with existing weakly supervised problems, such as semi-supervised classification, PU learning, and semi-supervised clustering, SU classification is a brand new problem. In semi-supervised learning, positive, negative, and unlabeled instances are available to train a binary classifier. The problem setting of semi-supervised classification seems to be similar to that of SU classification since the similar pairs are drawn from positive or negative instances. However, the real labels of instances in similar pairs set are unknown. In PU learning, only positive and unlabeled instances are available. Compared with SU classification, the problem setting in SU is much more complex since it contains pairwise information. Besides, the information available in SU classification is similar to semi-supervised clustering (Calandriello et al. 2014). However, the goals of these two problems are different. Specifically, the goal of SU classification is to minimize the risk function to train a classifier, which can be used to predict the unseen data, while semi-supervised clustering is used to

partition the data at hand. Recently, Lu et al. (2019, 2020) pointed out that PU learning and SU learning can be viewed as a special case of unlabeled-unlabeled (UU) learning and both of them suffer from over-fitting caused by negative loss. Great attention has been attracted to solve this problem in UU learning problem (Kiryo et al. 2017; Lu et al. 2020).

3 Preliminaries

In this section, we first give a brief review of binary supervised classification. Then, we give a brief review of SU classification and how to estimate the class-prior only use the similar unlabeled data. Finally, we give a brief review of RFF and DSG.

3.1 Supervised binary classification

Let $x_i \in \mathbb{R}^d$ be a d -dimensional data sample and $y_i \in \{-1, +1\}$ be the class label. Let the joint probability distribution density of the data instance $\{x_i, y_i\}$ be $p(x, y)$. In supervised binary classification problems, we need to minimize the following classification error to train a classifier $f : \mathbb{R}^d \mapsto \mathbb{R}$

$$R(f) = \mathbb{E}_{(x_i, y_i) \sim p(x, y)} [l(f(x_i), y_i)], \quad (1)$$

where $\mathbb{E}_{(x_i, y_i) \sim p(x, y)} [\cdot]$ denotes the expectation over the joint distribution density $p(x, y)$. $l(f(x_i), y_i)$ denotes the loss function which measures the difference between the prediction value $f(x_i)$ and the real labels y_i .

3.2 Similar unlabeled learning

To improve the performance of supervised learning methods, an effective way is to collect more labeled data. However, in real-world applications, labeling the data manually is time-consuming. Therefore, it is more practicable to directly use a large number of unlabeled samples to train the classifier. Besides, we may collect some implicit information, such as two data samples has the same label, but not the explicit labels. Bao et al. (2018) pointed out that we can use the unlabeled data samples together with the samples sharing the similar labels to train the classifier, which is denoted as ‘similar-unlabeled learning’. We will give a brief introduction of it in this subsection. We first discuss the underlying distributions of similar data pairs and unlabeled datasets based on Bao et al. (2018).

Similar Data Pairs Assume that any pair of instances $\{x, \hat{x}\}$ in the similar pairs set share the same but unknown label y . Then, they can be viewed as drawn from the following joint distribution density,

$$\begin{aligned} p_S(x, \hat{x}) &= p(x, \hat{x} | y = \hat{y} = 1 \cup y = \hat{y} = -1) \\ &= \frac{\pi_+^2 p_+(x) p_+(\hat{x}) + \pi_-^2 p_-(x) p_-(\hat{x})}{\pi_+^2 + \pi_-^2}, \end{aligned} \quad (2)$$

where $\pi_+ = p(y = 1)$ and $\pi_- = p(y = -1)$ denote the class prior and $\pi_+ + \pi_- = 1$. Besides, $p_+(x) = p(x | y = +1)$ and $p_-(x) = p(x | y = -1)$ denote the class conditional densities. Equation (2) means that the two data points x and $\hat{x}$ are drawn independently according

to the joint distribution density $p(x, y)$ and only the data pairs with the same labels will be accepted.

Unlabeled Dataset: Assume that the unlabeled samples are drawn according to the marginal density $p(x)$, where $p(x) = \pi_+ p_+(x) + \pi_- p_-(x)$.

To directly use the similar data pairs and the unlabeled data samples to train a classifier f , Bao et al. (2018) pointed out that we can minimize the following objective function,

$$R_{SU}(f) = \pi_S \mathbb{E}_{(x_i^s, \hat{x}_i^s) \sim p_S(x, \hat{x})} \left[\frac{\mathcal{L}_S(f(x_i^s)) + \mathcal{L}_S(f(\hat{x}_i^s))}{2} \right] + \mathbb{E}_{x_i^u \sim p(x)} [\mathcal{L}_U(f(x_i^u))], \quad (3)$$

where

$$\begin{aligned} \mathcal{L}_S(z) &= \frac{1}{2\pi_+ - 1} (l(z, +1) - l(z, -1)), \\ \mathcal{L}_U(z) &= -\frac{\pi_-}{2\pi_+ - 1} l(z, +1) + \frac{\pi_+}{2\pi_+ - 1} l(z, -1). \end{aligned}$$

and $\pi_S = \pi_+^2 + \pi_-^2$.

However, the expectation risk (3) cannot be directly minimized since the distribution of similar pairs set and unlabeled set are unknown. In practice, a common way to achieve the classifier is to use the empirical risk function to replace the expectation risk. Assume that there exists a similar pairs dataset $\mathcal{D}_S$ and an unlabeled dataset $\mathcal{D}_U$ defined as follows,

$$\begin{aligned} \mathcal{D}_S &= \{(x_i^s, \hat{x}_i^s)\}_{i=1}^{n_S} \stackrel{i.i.d.}{\sim} p_S(x, \hat{x}), \\ \mathcal{D}_U &= \{x_i^u\}_{i=1}^{n_U} \stackrel{i.i.d.}{\sim} p(x), \end{aligned}$$

where n_S and n_U denote the number of similar pairs and unlabeled data points, respectively. Then Equation (3) can be approximated by using the following empirical risk,

$$\hat{R}_S(f) = \frac{\pi_S}{n_S} \sum_{i=1}^{n_S} \frac{\mathcal{L}_S(f(x_i^s)) + \mathcal{L}_S(f(\hat{x}_i^s))}{2} + \frac{1}{n_U} \sum_{i=1}^{n_U} \mathcal{L}_U(f(x_i^u)) \quad (4)$$

In order to reduce the complexity of optimizing the pairwise problem, we can marginalize $p_S(x, \hat{x})$ with respect to $\hat{x}$ to decompose Equation (4) into a point-wise problem Bao et al. (2018). Assume that the similar pairs are independently drawn according to $p_S(x, \hat{x})$, then the marginalized distribution density of one of the data point x is:

$$\begin{aligned} \int p_S(x, \hat{x}) d\hat{x} &= \frac{\pi_+}{\pi_S} p_+(x) \int p_+(\hat{x}) d\hat{x} + \frac{\pi_-}{\pi_S} p_-(x) \int p_-(\hat{x}) d\hat{x} \\ &= \frac{\pi_+}{\pi_S} p_+(x) \int \frac{p(\hat{x}, y=+1)}{p(y=+1)} d\hat{x} + \frac{\pi_-}{\pi_S} p_-(x) \int \frac{p(\hat{x}, y=-1)}{p(y=-1)} d\hat{x} \\ &= \frac{\pi_+}{\pi_S} p_+(x) + \frac{\pi_-}{\pi_S} p_-(x) \\ &= \tilde{p}_S. \end{aligned} \quad (5)$$

This means that both x and $\hat{x}$ are drawn from the distribution density $\tilde{p}_S$. Thus, we can rewrite the the expectation objective function (3) as follows,

$$\begin{aligned}
R_{SU}(f) &= \pi_S \mathbb{E}_{(x_i^s, \hat{x}_i^s) \sim p_S(x, \hat{x})} \left[\frac{\mathcal{L}_S(f(x_i^s)) + \mathcal{L}_S(f(\hat{x}_i^s))}{2} \right] + \mathbb{E}_{x_i^u \sim p(x)} [\mathcal{L}_U(f(x_i^u))] \\
&= \pi_S \mathbb{E}_{(x_i^s, \hat{x}_i^s) \sim p_S(x, \hat{x})} \left[\frac{\mathcal{L}_S(f(x_i^s))}{2} \right] + \pi_S \mathbb{E}_{(x_i^s, \hat{x}_i^s) \sim p_S(x, \hat{x})} \left[\frac{\mathcal{L}_S(f(\hat{x}_i^s))}{2} \right] + \mathbb{E}_{x_i^u \sim p(x)} [\mathcal{L}_U(f(x_i^u))] \\
&= \pi_S \mathbb{E}_{x_i^s \sim \tilde{p}_S(x)} \left[\frac{\mathcal{L}_S(f(x_i^s))}{2} \right] + \pi_S \mathbb{E}_{\hat{x}_i^s \sim \tilde{p}_S(\hat{x})} \left[\frac{\mathcal{L}_S(f(\hat{x}_i^s))}{2} \right] + \mathbb{E}_{x_i^u \sim p(x)} [\mathcal{L}_U(f(x_i^u))] \\
&= \pi_S \mathbb{E}_{x_i^s \sim \tilde{p}_S(x)} [\mathcal{L}_S(f(x_i^s))] + \mathbb{E}_{x_i^u \sim p(x)} [\mathcal{L}_U(f(x_i^u))]
\end{aligned} \tag{6}$$

Similarly, we can use the empirical risk function to approximate the above objective function as follows,

$$\hat{R}_S(f) = \frac{\pi_S}{2n_S} \sum_{i=1}^{2n_S} \mathcal{L}_S(f(\tilde{x}_i^s)) + \frac{1}{n_U} \sum_{i=1}^{n_U} \mathcal{L}_U(f(x_i^u)), \tag{7}$$

and the similar pairs dataset $\mathcal{D}_S$ can be regarded as $\tilde{\mathcal{D}}_S = \{\tilde{x}_i\}_{i=1}^{2n_S} \stackrel{i.i.d.}{\sim} \tilde{p}_S$.

Recently, researchers pointed out that the SU problem is a special case in the unlabeled-unlabeled (UU) learning problem Lu et al. (2019) and the empirical risk function (7) of UU learning will go negative when using a complicated model leading to the model suffer from severe over-fitting Lu et al. (2020). To overcome this problem, Lu et al. (2020) propose to use the Consistent Correction Function $\delta(t)$ to correct the empirical risk estimation, where $\delta(t)$ is Lipschitz continuous, non-negative and $\delta(t) = t$ for all $t \geq 0$, such as absolute value function, ReLU function and generalized Leaky ReLU function. Therefore, the corrected risk estimation can be written as follows,

$$\begin{aligned}
\hat{R}_{cs}(f) &= \delta \left(\frac{\pi_S}{2n_S(2\pi_+ - 1)} \sum_{i=1}^{2n_S} l(f(\tilde{x}_i^s), 1) - \frac{\pi_-}{n_U(2\pi_+ - 1)} \sum_{i=1}^{n_U} l(f(x_i^u), 1) \right) \\
&\quad + \delta \left(\frac{\pi_+}{n_U(2\pi_+ - 1)} \sum_{i=1}^{n_U} l(f(x_i^u), -1) - \frac{\pi_S}{2n_S(2\pi_+ - 1)} \sum_{i=1}^{2n_S} l(f(\tilde{x}_i^s), -1) \right).
\end{aligned} \tag{8}$$

Note that the empirical risk function (7) could be convex if the loss function $l(z, t)$ is convex, twice differentiable in z almost everywhere (for every fixed $t \in \{\pm 1\}$), and satisfies the condition

$$l(z, +1) - l(z, -1) = -z.$$

However, once we add the Consistent Correction Functions, the new empirical risk function (8) is non-convex.

Instead of directly optimizing Equation (8), in real-world applications, a regularization term is added to restrict the solution space and penalize the complex solution. Then, we can minimize the following risk to train a classifier f in Reproducing Kernel Hilbert Space (RKHS) $\mathcal{H}$,

$$\begin{aligned}
J(f) &= \frac{\lambda}{2} \|f\|_{\mathcal{H}}^2 + \delta \left(\frac{\pi_S}{2n_S(2\pi_+ - 1)} \sum_{i=1}^{2n_S} l(f(\tilde{x}_i^s), 1) - \frac{\pi_-}{n_U(2\pi_+ - 1)} \sum_{i=1}^{n_U} l(f(x_i^u), 1) \right) \\
&\quad + \delta \left(\frac{\pi_+}{n_U(2\pi_+ - 1)} \sum_{i=1}^{n_U} l(f(x_i^u), -1) - \frac{\pi_S}{2n_S(2\pi_+ - 1)} l(f(\tilde{x}_i^s), -1) \right),
\end{aligned} \tag{9}$$

where $\lambda > 0$ denotes the regularization parameter and $\|\cdot\|_{\mathcal{H}}$ denotes the norm in RKHS.

3.3 Class-prior estimation

In the above subsection, we assume that the class prior is already known. However, in many real-world applications, the positive class prior is not given. To solve this problem, Bao et al. (2018) estimate the class prior from the similar-unlabeled data. Specifically, assume that we have two samples x and $\hat{x}$ independently drawn from the same dataset, such that have

$$\begin{aligned} p(x, \hat{x}) &= p(x)p(\hat{x}) \\ &= \pi_+^2 p_+(x)p_+(\hat{x}) + \pi_-^2 p_-(x)p_-(\hat{x}) + \pi_+ \pi_- p_+(x)p_-(\hat{x}) + \pi_+ \pi_- p_-(x)p_+(\hat{x}) \\ &= \pi_S p_S(x, \hat{x}) + \pi_D p_D(x, \hat{x}) \end{aligned} \quad (10)$$

where $\pi_D := 2\pi_+ \pi_-$ and

$$\begin{aligned} p_D(x, \hat{x}) &= p(x, \hat{x}) | (y = +1 \wedge \hat{y} = -1) \vee (y = -1 \wedge \hat{y} = +1) \\ &= \frac{\pi_+ \pi_- p_+(x)p_-(\hat{x}) + \pi_+ \pi_- p_-(x)p_+(\hat{x})}{2\pi_+ \pi_-}. \end{aligned} \quad (11)$$

Then, we can obtain

$$p(x) = \pi_S \tilde{p}_S(x) + \pi_D \tilde{p}_D(x) \quad (12)$$

and $\tilde{p}_D(x) := \frac{p_+(x) + p_-(x)}{2}$. Since the similar data pairs and unlabeled samples are drawn from $\tilde{p}_S(x)$ and $p(x)$, we can use mixture proportion estimation methods (Scott 2015; Sakai et al. 2017; Ramaswamy et al. 2016) to estimate π_S . After estimating π_S , we can calculate π_+ by using the equation $\pi_+ = \frac{\sqrt{2\pi_S - 1} + 1}{2}$. (Note here we assume $\pi_+ > \pi_-$)

3.4 Random fourier features

In this subsection, we briefly introduce Random Fourier Features (RFF). According to the Bonchner' Theorem (Rakhlin et al. 2012), for any *continuous, real-valued, symmetric* and *shift-invariant* kernel function $k(x, x')$, (such as Gaussian kernel $k(x, x') = \exp(-\sigma \|x - x'\|_2^2)$, Laplacian kernel $k(x, x') = \exp(-\sigma \|x - x'\|_1)$ and Matérn kernel Smola and Schölkopf 1998) there exists a non-negative Fourier transform function, $k(x, x') = \int_{\mathbb{R}^d} p(\omega) e^{j\omega^T(x-x')} d\omega$. If we regard $p(\omega)$ as a density function, the kernel function $k(x, x')$ is equal to the expectation of $e^{j\omega^T(x-x')}$ over $p(\omega)$. Besides, the integrand $e^{j\omega^T(x-x')}$ can be replaced with $\cos \omega^T(x - x')$ Rahimi and Recht (2008). Then we can obtain a real-valued feature mapping $\phi_\omega(x) = [\cos(\omega^T x), \sin(\omega^T x)]^T$, where ω is a randomly sampled D -dimensional vector according to the density function $p(\omega)$. Obviously, we have $E[\phi_\omega^T(x) \phi_\omega(x')] = k(x, x')$ which means the feature mapping is an unbiased estimation of kernel function. In order to achieve a low variance approximation of $k(x, x')$, the feature mapping $\phi_w(x)$ can be written as follows,

$$\phi_\omega(x) = \sqrt{1/D} [\cos(\omega_1^T x), \dots, \cos(\omega_m^T x), \sin(\omega_1^T x), \dots, \sin(\omega_m^T x)]^T. \quad (13)$$

With this feature mapping (13), the infinite-dimensional $f(\cdot)$ can be approximated by

$$f(\cdot) = \sum_{i=1}^n \beta_i k(x_i, \cdot) = \sum_{i=1}^n \beta_i \phi_\omega(x_i) \phi_\omega(\cdot), \quad (14)$$

where $\{\beta_i\}_{i=1}^n$ denotes the weights assigned to each training instance.

Explicit random features for many other kernels have been derived (Dai et al. 2014), such as polynomial kernels $k(x, x') = (a\langle x, x' \rangle + c)^p$ Pham and Pagh (2013), additive/multiplicative class of homogeneous kernels Vedaldi and Zisserman (2012) and Intersection kernel Yang et al. (2014).

3.5 Doubly stochastic gradient method

In this subsection, we give a brief review of Doubly Stochastic Gradient method (DSG). The goal of DSG is to learn a classifier $f \in \mathcal{H}$ by minimizing

$$\tilde{\mathcal{R}}(f) = \mathbb{E}_{(x,y) \sim p(x,y)} [l(f(x), y)] + \frac{\lambda}{2} \|f\|_{\mathcal{H}}^2. \quad (15)$$

In each update iteration, DSG randomly sample a data point and its random features to calculate the approximate functional gradient as follows,

$$\zeta(\cdot) = l'(f(x_i), y) \phi_\omega(x_i) \phi_\omega(\cdot) \quad (16)$$

Then, we can use this approximate functional gradient to update the model function. In order to avoid keeping all the random features in memory, DSG saves a seed of the random number generator to regenerate the random features on-the-fly. Dai et al. (2014) also prove that DSG has a convergence rate of $O(1/T)$. However, their convergence analysis is based on the convexity assumptions of the objective function. These assumptions are restrictive in real-world applications.

4 Triply stochastic gradient method for similar-unlabeled learning

In this section, we propose our TSGSU algorithm for large-scale SU classification problems with kernels.

4.1 Triply stochastic gradient method

A common method to optimize the objective function (9) is to use gradient descent methods. Based on the definition of the function $f \in \mathcal{H}$, we can easily obtain $\nabla f(x) = \nabla \langle f, k(x, \cdot) \rangle_{\mathcal{H}} = k(x, \cdot)$, and $\nabla \|f\|_{\mathcal{H}}^2 = \nabla \langle f, f \rangle_{\mathcal{H}} = 2f$. Thus, the full gradient of the objective function (9) is

$$\begin{aligned} \nabla J(f) = & \lambda f + \delta'_1 \cdot \left(\frac{\pi_S}{2n_S(2\pi_+ - 1)} \sum_{i=1}^{2n_S} l'(f(\tilde{x}_i), 1) k(\tilde{x}_i, \cdot) - \frac{\pi_-}{n_U(2\pi_+ - 1)} \sum_{i=1}^{n_U} l'(f(x_i), 1) k(x_i, \cdot) \right) \\ & + \delta'_2 \cdot \left(\frac{\pi_+}{n_U(2\pi_+ - 1)} \sum_{i=1}^{n_U} l'(f(x_i), -1) k(x_i, \cdot) - \frac{\pi_S}{2n_S(2\pi_+ - 1)} \sum_{i=1}^{2n_S} l'(f(\tilde{x}_i), -1) k(\tilde{x}_i, \cdot) \right), \end{aligned} \quad (17)$$

where $l'(f(x_i), \cdot) k(x_i, \cdot)$ is the derivative of $l(f(x_i), \cdot)$ w.r.t f and δ' is the derivative of $\delta(\cdot)$.

Stochastic Functional Gradient Directly calculating the full gradient (17) is time-consuming for large-scale problems. To reduce the computational complexity, we can update the model function in a stochastic framework. Considering there exist two data sources in SU classification, *i.e.*, similar pairs dataset and unlabeled dataset, we introduce another layer of random sampling into the standard DSG framework to obtain the stochastic functional gradients. Specifically, we randomly sample two data points, *i.e.*, $\tilde{x}_t^s$ and x_t^u , where $\tilde{x}_t^s$ is randomly sampled from the similar pairs dataset $\tilde{\mathcal{D}}_S$ and x_t^u is randomly sampled from the unlabeled dataset $\mathcal{D}_U$ in each iteration. Then the stochastic functional gradient of Equation (8) can be easily obtained,

$$\begin{aligned} \xi(\cdot) = & \delta'_1 \cdot \left(\frac{\pi_S}{2\pi_+ - 1} l'(f(\tilde{x}_t^s), 1)k(\tilde{x}_t^s, \cdot) - \frac{\pi_-}{2\pi_+ - 1} l'(f(x_t^u), 1)k(x_t^u, \cdot) \right) \\ & + \delta'_2 \cdot \left(\frac{\pi_+}{2\pi_+ - 1} l'(f(x_t^u), -1)k(x_t^u, \cdot) - \frac{\pi_S}{2\pi_+ - 1} l'(f(\tilde{x}_t^s), -1)k(\tilde{x}_t^s, \cdot) \right). \end{aligned} \quad (18)$$

Random Feature Approximation To further reduce the complexity of explicitly calculating the kernel function $k(x, \cdot)$ in gradient (18), we can apply random Fourier features to further approximate the functional gradient. Then we can achieve the following functional gradient,

$$\begin{aligned} \zeta(\cdot) = & \delta'_1 \cdot \left(\frac{\pi_S}{2\pi_+ - 1} l'(f(\tilde{x}_t^s), 1)\phi_\omega(\tilde{x}_t^s)\phi_\omega(\cdot) - \frac{\pi_-}{2\pi_+ - 1} l'(f(x_t^u), 1)\phi_\omega(x_t^u)\phi_\omega(\cdot) \right) \\ & + \delta'_2 \cdot \left(\frac{\pi_+}{2\pi_+ - 1} l'(f(x_t^u), -1)\phi_\omega(x_t^u)\phi_\omega(\cdot) - \frac{\pi_S}{2\pi_+ - 1} l'(f(\tilde{x}_t^s), -1)\phi_\omega(\tilde{x}_t^s)\phi_\omega(\cdot) \right). \end{aligned} \quad (19)$$

Since we randomly sample three variables, *i.e.*, $\tilde{x}_t^s$, x_t^u and w , we can call our functional gradient (19) as triply stochastic functional gradient.

4.2 Updating rules

Now we propose the update rules of our proposed TSGSU. We first present the unbiased estimation of the full gradient (17) by using either $\xi(\cdot)$ or $\zeta(\cdot)$ as follows,

$$\nabla J(h) = \lambda h + \mathbb{E}_{\tilde{x}^s}[\mathbb{E}_{x^u}[\xi(\cdot)]], \quad (20)$$

$$\nabla J(f) = \lambda f + \mathbb{E}_{\tilde{x}^s}[\mathbb{E}_{x^u}[\zeta(\cdot)]]. \quad (21)$$

For convenience, the function value is expressed as $h(x)$ if updated by using gradient (20), and is expressed as $f(x)$ if updated by using gradient (21). Obviously, $h(x)$ is always in the RKHS $\mathcal{H}$ since it is updated by using real functional gradient while $f(x)$ may be outside the RKHS $\mathcal{H}$.

Let $h_1(\cdot) = f_1(\cdot) = 0$. Then, we present the update rules using the true stochastic functional gradient $\xi(\cdot)$ at t -th iteration as follow,

$$h_{t+1}(\cdot) = h_t(\cdot) - \eta_t(\xi_t(\cdot)) + \lambda h_t(\cdot) = \sum_{i=1}^t \alpha_t^i \xi_t^i(\cdot), \quad \forall t > 1, \quad (22)$$

where $a_t^i = -\eta_i \prod_{j=i+1}^t (1 - \eta_j \lambda)$, and η_t denotes the step size. Since $\zeta(\cdot)$ is an unbiased estimation of $\xi(\cdot)$, the update rule by using $\zeta(\cdot)$ is similar to that by using $\xi(\cdot)$. So the update rule at t -th iteration by using $\zeta(\cdot)$ is

$$f_{t+1}(\cdot) = f_t(\cdot) - \eta_t(\zeta_t(\cdot) + \lambda f_t(\cdot)) = \sum_{i=1}^t a_t^i \zeta_i(\cdot), \quad \forall t > 1, \quad (23)$$

However, using $\zeta_t(\cdot)$ and $\xi_t(\cdot)$ to update data the model function, we still need to calculate the infinite-dimensional feature map. To further speed up our algorithm, we introduce a sequence of constantly-changing coefficients $\{\alpha_t\}_{t=1}^T$ into the computation of the model function. Then the update rules can be rewritten as follows,

$$f_t = \sum_{i=1}^t \alpha_i \phi_\omega(x), \quad (24)$$

$$\begin{aligned} \alpha_t = & -\eta_t(\delta'_1 \cdot \left(\frac{\pi_S}{2\pi_+ - 1} l'(f(\tilde{x}_t^s), 1) \phi_\omega(\tilde{x}_t^s) - \frac{\pi_-}{2\pi_+ - 1} l'(f(x_t^u), 1) \phi_\omega(x_t^u) \right) \\ & + \delta'_2 \cdot \left(\frac{\pi_+}{2\pi_+ - 1} l'(f(x_t^u), -1) \phi_\omega(x_t^u) - \frac{\pi_S}{2\pi_+ - 1} l'(f(\tilde{x}_t^s), -1) \phi_\omega(\tilde{x}_t^s) \right)), \end{aligned} \quad (25)$$

$$\alpha_j = (1 - \eta_t \lambda) \alpha_j, \quad \text{for } j = 1, \dots, t-1. \quad (26)$$

4.3 TSGSU algorithms

We present the overall algorithms in Algorithm 1 and 2. In order to achieve a good generalization performance by using random features, large amounts of random features need to be saved in the memory. So it is necessary to design an efficient method to generate random features. Fortunately, according to Dai et al. (2014), we can use a *pseudo-random number generator* with seed i to generate random features in each iteration, instead of saving all the random features. With this seed i aligned between the training and prediction process, we can guarantee that the random features generated during the training and prediction process are the same. For faster implementation, TSGSU keeps updating the sequence of coefficient α_j , which represents the weights of the random features generated so far. In each iteration of the training process, the following steps need to be executed:

Algorithm 1 $\{\alpha_i\}_{i=1}^t = \text{TSGSU}(\tilde{x}_t^s, x_t^u, p(\omega))$

Input: $p(\omega)$, $\phi_\omega(\cdot)$, λ .**Output:** $\{\alpha_t\}_{t=1}^T$

```

1: for  $t = 1, \dots, T$  do
2:   Sample  $\tilde{x}_t$  from  $\tilde{\mathcal{D}}_S$ .
3:   Sample  $x_t^u$  from  $\mathcal{D}_U$ .
4:   Sample  $\omega_t \sim p(\omega)$  with seed  $t$ .
5:    $f(x_t) = \text{Predict}(x, \{\alpha_j\}_{j=1}^{t-1})$ .
6:    $\alpha_t = -\eta_t(\delta'_1 \cdot \left( \frac{\pi_S}{2\pi_+ - 1} l'(f(\tilde{x}_t^s), 1)\phi_\omega(\tilde{x}_t^s) - \frac{\pi_-}{2\pi_+ - 1} l'(f(x_t^u), 1)\phi_\omega(x_t^u) \right) + \delta'_2 \cdot$ 
       $\left( \frac{\pi_+}{2\pi_+ - 1} l'(f(x_t^u), -1)\phi_\omega(x_t^u) - \frac{\pi_S}{2\pi_+ - 1} l'(f(\tilde{x}_t^s), -1)\phi_\omega(\tilde{x}_t^s) \right)),$ 
7:    $\alpha_j = (1 - \eta_t\lambda)\alpha_j$ , for  $j = 1, \dots, t-1$ .
8: end for
```

Algorithm 2 $f(x) = \text{Predict}(x, \{\alpha_i\}_{i=1}^t)$

Input: $p(\omega)$, $\phi_\omega(\cdot)$ **Output:** $f(x)$

```

1: Set  $f(x) = 0$ .
2: for  $i = 1, \dots, t$  do
3:   Sample  $\omega_i \sim p(\omega)$  with seed  $i$ .
4:    $f(x) = f(x) + \alpha_i \phi_{\omega_i}(x)$ 
5: end for
```

1. *Select random data pairs.* Construct data pairs by randomly sampling two instances from the similar set $\tilde{\mathcal{D}}_S$ and unlabeled set $\mathcal{D}_U$, respectively. Here a mini-batch can be used to further improve our method.
2. *Generate random features.* Use a pseudo-random number generator with seed i to sample w_i and calculate random features.
3. *Compute the function value.* Calculate the function value $f(x)$ according to Equation (20).
4. *Update the constantly-changing coefficients.* Compute the current t -th iteration coefficients α_t and then update the previous coefficients α_j , $j = 1, \dots, t-1$ according update rules.

5 Convergence analysis

In this section, we will show the convergence rate of TSGSU. We first give several assumptions used in our analysis:

Assumption 1 (Bound of kernel function.) Let $\kappa > 0$, such that $k(x, x') \leq \kappa$.

Assumption 2 (Bound of random feature norm.) Let $\phi > 0$, such that $|\phi_\omega(x)\phi_\omega(x')| \leq \phi$.

Assumption 3 (Bound of derivatives.) The derivative of l w.r.t the first term u is bounded: $|l'(u, v)| < M$ and the derivative of δ is bounded: $|\delta'| < M_\delta$.

Assumption 4 (Lipschitz continuous.) $l(u, v)$ is L -Lipschitz continuous.

Assumption 5 (Lipschitz gradient.) The gradient function $\nabla J(f)$ is Lipschitz continuous such that

$$\|\nabla J(f) - \nabla J(g)\|_{\mathcal{H}} \leq L_j \|f - g\|_{\mathcal{H}}, \quad \forall f, g \in \mathcal{H}$$

Assumption 6 The objective function $J(h_t)$ is bounded below by a scalar $J_{\inf}$

Let χ denote the whole training set. Obviously, since the objective function could be non-convex, our purpose of the convergence analysis is to bound $\mathbb{E}[\|\nabla J(f)\|_2^2]$ Geng et al. (2019). However, since we use the approximate gradient to update f , f may be outside the RKHS $\mathcal{H}$. Therefore, instead of bound $\mathbb{E}[\|\nabla J(f)\|_2^2]$, we first bound the gradient of the objective function using the exact kernel function, i.e., $\mathbb{E}[\|\nabla J(h)\|_2^2] < \epsilon$. Then we prove that f is close to h at the stationary point.

Here, we first give the error bound caused by random features in the following Lemma.

Lemma 1 $\forall x \in \chi$, for a fixed step size $\eta_t = \bar{\eta} = \frac{\theta}{\sqrt{T}}$, $0 < \theta \leq 1$, we have that

$$\mathbb{E}_{\tilde{x}^S, x^u, \omega} [|f_{T+1}(x) - h_{T+1}(x)|^2] \leq B_{1,T+1}^2,$$

where $B_{1,T+1}^2 := (\frac{2\pi_S + 1}{2\pi_+ - 1})^2 M_\delta^2 M^2 (\sqrt{\kappa} + \sqrt{\phi})^2 \theta^2$ and $\mathbb{E}_{\tilde{x}^S, x^u, \omega}[\cdot]$ denotes the expectation over the similar dataset, unlabeled dataset and the random features.

Remark 1 According to Lemma 1, we have that for an appropriate step size, we can easily bound the error caused by randomly sampled random features by a constant value.

Then, we bound gradient of the objective function updated by using the true functional gradient.

Theorem 1 (Convergence in expectation) For a fixed step size $\eta_t = \bar{\eta} = \frac{\theta}{\sqrt{T}}$, $0 < T \leq 1$, we have that

$$\begin{aligned} & \mathbb{E} \left[\frac{1}{T} \sum_{t=1}^T \|\nabla J(h_t)\|_{\mathcal{H}}^2 \right] \\ & \leq \frac{J(h_1) - J_{\inf}}{\theta \sqrt{T}} + (1 + \lambda) \left(\frac{2\pi_S + 1}{2\pi_+ - 1} \right)^3 M_\delta^3 M^2 L_K (\sqrt{\kappa} + \sqrt{\phi}) \frac{\theta^2}{\sqrt{T}} + 2L_j \frac{\theta}{\sqrt{T}} \left(\frac{\pi_S + 1}{2\pi_+ - 1} \right)^2 M_\delta^2 M^2 \kappa \end{aligned} \quad (27)$$

Remark 2 From Theorem 1, we can obtain that for any given data point x , our TSGSU can convergence to the stationary point at the rate of $O(\frac{1}{\sqrt{T}})$ for an appropriate step size.

6 Experiments

In this section, we conduct experiments on various datasets to verify the effectiveness and efficiency of TSGSU.

6.1 Experimental setups

We compare the running time and scalability of TSGSU with existing SU algorithms summarized as follows,

1. **SU-SL** The method proposed in Bao et al. (2018) which directly optimize the objective function (7) with squared loss $l(u, v) = \frac{1}{4}(uv - 1)^2$. When using this loss function, the objective function (7) becomes convex and the optimal solution can be analytically obtained. We use Python code from https://github.com/levelfour/SU_Classification as the implementation of SU-SL.
2. **SU-DH** The method proposed in Bao et al. (2018) directly optimize the objective function (7) with doubly hinge loss $l(u, v) = \max\{-uv, \max\{0, \frac{1}{2} - \frac{1}{2}uv\}\}$. In this case, objective function is convex, and the convex optimization methods can be used to solve this problem, such as the *cvxopt* Python package. We use Python code from https://github.com/levelfour/SU_Classification as the implementation of SU-DH.
3. **CUU** The method proposed in Lu et al. (2020). It uses a stochastic gradient method to solve the objective function (8). Specifically, in each iteration, it randomly samples a batch of unlabeled data points and a batch of similar labeled data points and calculates their kernel function. Then the gradient method can be used to update the model parameters. In our Python implementation, we use the squared loss $l(u, v) = \frac{1}{4}(uv - 1)^2$ and absolute value function to avoid the negative loss.
4. **TSGSU** Our proposed large-scale SU classification algorithm based on the doubly stochastic gradient framework. Specifically, we first randomly sample a batch of unlabeled data points and a batch of similar labeled data points. Instead of directly calculating the kernel function, we use the Random Fourier features to approximate the kernel. Then, we can obtain the approximate functional gradient and use it to update the model function. In our Python implementation, we use the squared loss $l(u, v) = \frac{1}{4}(uv - 1)^2$ and absolute value function to avoid the negative loss.

Note that all the experiments are run on a PC with 56 2.2Ghz cores and 80GB RAM. All the experiments are ran for 10 trials. For all experiments, the positive class prior π_+ is given by the proportion of positive samples in the datasets. Besides, it can be estimated from the SU data by using the method introduced in Section 3.3.

6.2 Datasets

In order to demonstrate the superiority of our proposed method, we conduct the experiments on several benchmark datasets and high dimensional datasets.

Table 2 Benchmark datasets

Dataset	Features	Samples	Source
Acoustic	50	98,528	LIBSVM
Combined	100	98,528	LIBSVM
Poker	10	11,000,000	LIBSVM
Year	90	463,715	UCI
Covtype	54	581,012	UCI
Susy	18	5,000,000	UCI

Table 3 High dimensional datasets

Dataset	Features	Samples	Source
Mnist8m	784	8,100,000	LIBSVM
Sim	2000	72,309	LIBSVM
Svhn	3072	99,289	LIBSVM

Benchmark datasets We choose six large-scale datasets collected from LIBSVM¹ and UCI² repositories, which are summarized in Table 2. Originally, these datasets are used for supervised multi-class classification. To conduct the experiments of SU classification, we need to transfer these fully labeled datasets to partially labeled datasets and we choose a single label to be positive and the rest labels to be negative. Besides, we double the size of Acoustic and Combined by adding a random noise in the first dimension. Then, all these datasets are decomposed into three parts, *i.e.*, similar pairs set, unlabeled dataset and test dataset. Specifically, we randomly sample 5000 similar pairs and $n_U = \{10000, 100000\}$ instances as unlabeled by dropping their labels according to the class-prior. Then $n_U/3$ test instances are randomly sampled from the rest data points.

High dimensional datasets Since high dimensional datasets widely exist in real-world applications, we also conduct extensive experiments on high dimensional datasets, which are summarized in Table 3. Originally, Mnist8m and Svhn have 10 classes. We need to transfer them to binary versions. For Mnist8m, we choose a single label as positive and regard the rest as negative in each time. For Svhn, we classify digits 0 to 4 versus 5 to 9. In addition, We double the size of Sim and Svhn by adding a random noise in the first dimension. Then we can use the same way mentioned above to generate SU data.

For the benchmark datasets, we use the Gaussian kernel $k(x, x') = \exp(-\sigma \|x - x'\|^2)$ to build the non-linear model in SUSL, SUDH and TSGSU. Besides, in the CUU, we use the basic function $\phi(x) = \exp(-\sigma \|x - x'\|^2)$ to build the non-linear model, where x' denotes the kernel center. Note that a common method to speed up the CUU is to randomly sample several points as the kernel centers. Thus, in our experiments, we first randomly sample 2000 points as the centers in CUU and we denote it as CUU(2000). Then, we use all training data as the kernel centers and we denote it as CUU(All). The hyper-parameters λ and σ are chosen from 5-fold cross-validation from the range $\{(\lambda, \sigma) | 2^{-7} \leq \lambda \leq 2^7, 2^{-7} \leq \sigma \leq 2^7\}$. Specifically, we divide the similar pairs set and unlabeled data evenly into 5 parts respectively.

¹ <https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/binary/>

² <http://archive.ics.uci.edu/ml/datasets.html>

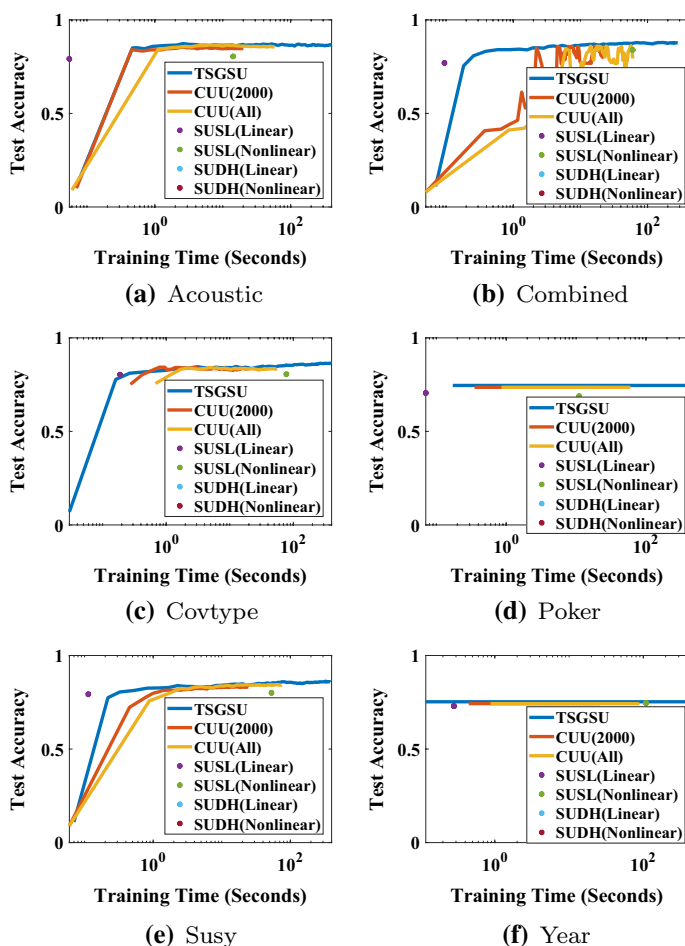


Fig. 1 The training time versus test accuracy of SUSL(Linear), SUSL(Nonlinear), SUDH(Linear), SUDH(Nonlinear), CUU(All), CUU(2000) and TSGSU against 10, 000 unlabeled samples on benchmark datasets, where the sizes of similar pairs are fixed at 5000. The non-linear model is built by Gaussian kernel. (The results of SUDH(Linear) and SUDH(Nonlinear) are not presented since the training time is larger than 5000 seconds. Since SUSL is not a iterative method, the results of SUSL are presented as points)

Then we do a grid search to choose the best hyper-parameters. Besides, we adjust the initial step size in TSGSU and CUU to achieve a better convergence performance. We fix the batch size of similar labeled data and unlabeled data both at 1000 and the iteration number at 100 for TSGSU and CUU to ensure that we can traverse all the training data at least one time. The number of random features of our TSGSU is fixed in 1000.

We present all the results of SUSL, SUDH, CUU, and TSGSU with 10,000 unlabeled samples in Table 4. We also present the results of SUSL and SUDH with the linear model in Table 4. We can find that all the results of the non-linear model are higher than those of the linear model. It highly demonstrates the superiority of the kernel method in dealing with the data with non-linear structures. Besides, we can also find that CUU and TSGSU have higher results in most cases than SUDH and SUSL when using the Gaussian kernel, i.e., Acoustic, Combined, Covtype, Poker, and Year. This is because that when dealing with

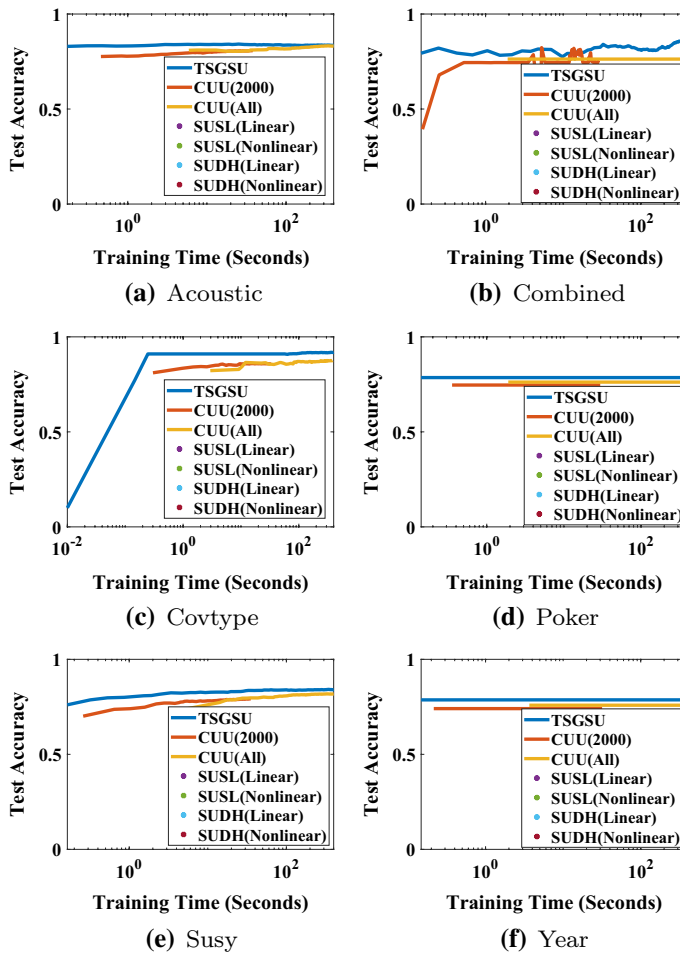


Fig. 2 The training time versus test accuracy of SUSL(Linear), SUSL(Nonlinear), SUDH(Linear), SUDH(Nonlinear), CUU(All), CUU(2000) and TSGSU against 100, 000 unlabeled samples on benchmark datasets, where the sizes of similar pairs are fixed at 5000. The non-linear model is built by Gaussian kernel. (The results of SUDH and SUSL are missing since they are out of memory)

the complex model, SUDH and SUSL will suffer from the negative loss and lead to overfitting. However, CUU and our TSGSU use the Correct Consistent Function to overcome this negative loss. We also present the training time of each dataset with 10,000 unlabeled samples in Fig. 1. We can find that the results of SUDH(Linear) and SUDH(Nonlinear) are missing since they need more than 5,000 seconds. This is because SUDH needs to maintain the kernel matrix with $O(n^3)$ operations and the cvxopt package cannot efficiently deal with it. Besides, since SUSL is not an iterative method, our method can generate the model faster even if it has a smaller total training time. It is also obvious that CUU is faster than our TSGSU in the small set. This is because, on such a small set, the number x' in the kernel function is not very large, and such that we can efficiently calculate the kernel function of the batch data. However, for our method, the main computational complexity is located in the iteration number since our method has $O(DT^2)$ computational complexity.

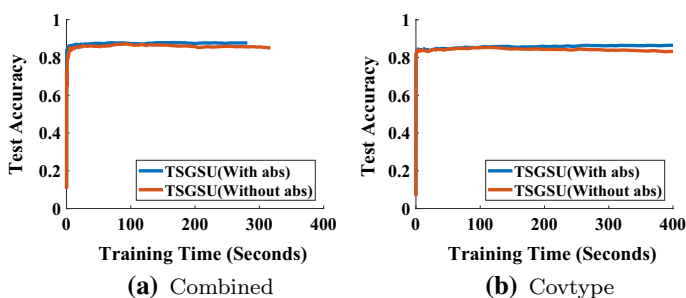


Fig. 3 The training time versus test accuracy of TSGSU with the absolute value function and without the absolute function

Table 4 The test accuracy (%) of SU-DH, SU-SL, CUU, and TSG on the high dimensional dataset with 5000 similar pairs and 10000 unlabeled data points. The non-linear model is built by the Gaussian kernel

Methods	Acoustic	Combined	Covtype	Poker	Susy	Year
SUSL(Linear)	79.61 ± 1.11	78.03 ± 0.98	80.63 ± 0.87	69.81 ± 0.63	79.61 ± 0.68	72.49 ± 0.83
SUSL(Nonlinear)	82.43 ± 0.85	83.83 ± 0.81	81.27 ± 0.57	70.05 ± 0.45	81.22 ± 0.67	73.46 ± 0.93
SUDH(Linear)	79.19 ± 0.93	78.91 ± 0.97	81.01 ± 0.63	69.97 ± 0.31	80.03 ± 0.38	73.05 ± 0.58
SUDH(Nonlinear)	81.91 ± 0.63	83.33 ± 0.92	81.35 ± 0.77	70.05 ± 0.45	81.24 ± 0.44	73.67 ± 0.69
CUU(2000)	81.91 ± 1.61	82.16 ± 0.97	80.67 ± 1.04	70.01 ± 0.35	80.17 ± 1.03	73.23 ± 0.94
CUU(All)	82.57 ± 0.15	83.83 ± 0.38	81.33 ± 0.76	70.05 ± 0.41	81.21 ± 0.65	73.49 ± 0.59
TSGSU	82.89 ± 0.69	83.88 ± 0.88	81.39 ± 0.79	70.09 ± 0.39	81.23 ± 0.66	73.49 ± 0.55

The best results are shown in bold

We also present the test accuracy versus training time for 100,000 unlabeled data points in Fig. 2. We can find that the results of SUSL and SUDH are all missing. This is because that they need more than $O(n^2)$ space and it is easy to out of the memory when the total number of the training sets is large enough. We can also find that our method has a faster convergence rate than that of CUU(All) in these large-scale datasets. This is because our method only needs to calculate the random features of the randomly sampled data in each iteration while CUU(All) needs to calculate the corresponding kernel function. When the number of x' becomes very large, calculating the kernel function of a batch data still needs a lot of time. Obviously, using 2000 points the kernel centers can highly speed up the CUU method. However, we can find that our method can generate the model faster than CUU(2000) and CUU(All). Besides, our method has a faster convergence rate than CUU(2000) and CUU(All). This is because that for the first few iterations, the time complexity of our method is much lower than that of CUU. Besides, we can find that our method has 1% higher accuracy than that of CUU(2000) in most cases. Considering the convergence speed and accuracy together, our proposed method is superior to CUU(2000).

Besides, we compare the performance of our method using the absolute value function and without the correct function on Combined and Covtype. The results are presented in Fig. 3. We can that without using the absolute function, there is a reduction in the test accuracy. This could be caused by the negative loss of the empirical risk function, and we can use the absolute function to avoid this problem.

Table 5 The test accuracy of SU-DH, SU-SL, CUU, and TSG on the high dimensional dataset with 5000 similar pairs and 10000 unlabeled data points. The non-linear model is built by the dot product kernel

Methods	Mnist 2	Mnist 4	Mnist 6	Mnist 8	Sim	Svhn
SUSL(Linear)	89.23 ± 0.54	88.99 ± 0.62	88.41 ± 0.28	88.33 ± 0.25	61.23 ± 0.87	62.31 ± 0.98
SUSL(Kernel)	89.93 ± 1.26	89.72 ± 2.12	89.61 ± 0.81	88.82 ± 0.77	62.78 ± 1.81	62.61 ± 1.81
SUDH(Linear)	89.49 ± 1.25	88.54 ± 0.86	89.43 ± 0.45	89.32 ± 0.45	65.17 ± 0.28	63.84 ± 0.24
SUDH(Kernel)	90.09 ± 1.79	89.72 ± 1.79	89.60 ± 0.81	90.40 ± 0.15	66.07 ± 0.56	64.07 ± 0.56
CUU(2000)	90.02 ± 0.78	89.88 ± 0.43	89.79 ± 0.33	89.78 ± 0.92	67.39 ± 0.41	64.20 ± 0.42
CUU(All)	90.19 ± 0.49	90.19 ± 0.38	90.96 ± 0.82	90.31 ± 1.05	68.02 ± 0.42	64.25 ± 0.75
TSGSU	90.26 ± 0.42	90.60 ± 0.37	90.10 ± 0.67	90.45 ± 0.97	68.31 ± 0.49	64.26 ± 0.75

The best results are shown in bold

We also evaluate the performance of TSGSU, CUU, SUSL and SUDH in high-dimensional datasets using the dot product $k(x, x') = (\sigma_1 \langle x, x' \rangle)^{\sigma_2}$. Similar to the previous experimental setups, we first evaluate the performance of the small subset of high-dimensional datasets. The hyper-parameters of σ_1 and σ_2 are chosen from $\{1, 0.1, 0.01, 0.001\}$ and $\{1, 3, 5\}$ by using the grid search. The test accuracy of TSGSU, CUU, SUSL, and SUDH of the nonlinear model is presented in Table 5. Besides, we also give the test accuracy of SUSL and SUDH with the linear model and the test accuracy of CUU with 2000 kernel centers in Table 5. From Table 5, we can find that compared with SUSL and CUU with dot product kernel, the results of our method are still the best in most cases. We also conduct our experiments on the large subset and we present the results in Fig. 4. Comparing the results in Fig. 4, we can find that with the increase of training size, SUSL and SUDH is out of the memory for the huge space complexity. Besides, our method has a faster convergence rate than that of CUU (All) since we only need to calculate random features instead of calculating the kernel function. This demonstrates that our method can be used in other kernel functions and it is still time-consuming. Besides, we also use 2000 points as the kernel centers in CUU. We can find it has a smaller training time than our method but a lower test accuracy.

Based on all these results, we can conclude that TSGSU is superior to other existing SU classification algorithms with kernels in terms of efficiency and scalability while retaining similar generalization performance.

7 Detailed proofs of convergence rate

In this section, we give detailed proofs of Lemma 1 and Theorem 1.

7.1 Proof of Lemma 1

Here we give the detailed proof of Lemma 1.

Proof We denote $A_t(x) = A_t(x; \tilde{x}_t^y, x_t^u, \omega_t) := a_t^i(\zeta_t(x) - \xi_t(x))$ for the t -th iteration. According to the assumption in section 5, $A_t(X)$ have a bound:

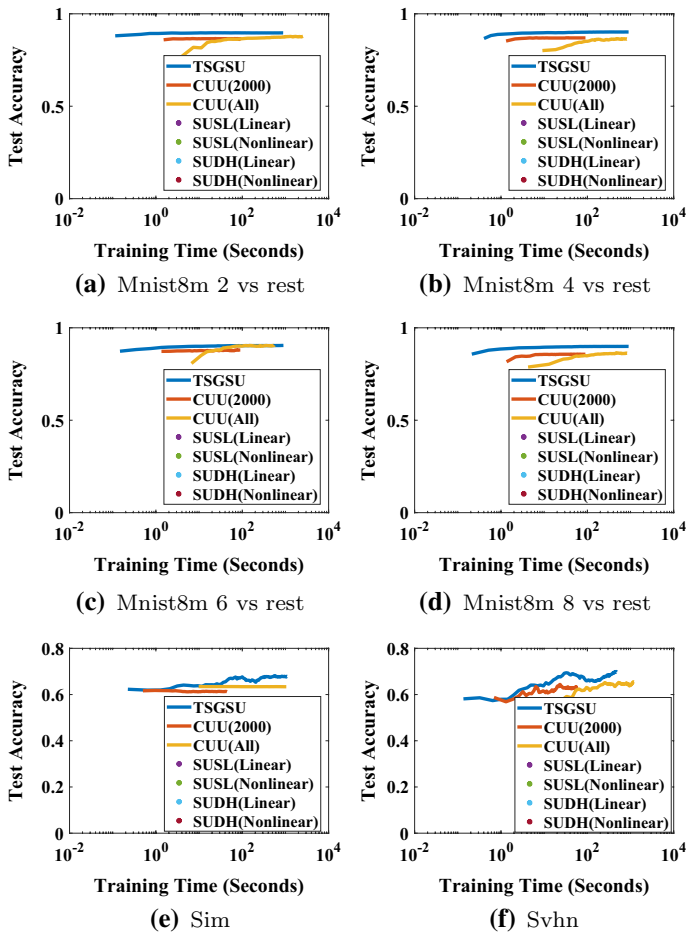


Fig. 4 The training time versus test accuracy of SUSL(Linear), SUSL(Nonlinear), SUDH(Linear), SUDH(Nonlinear), CUU(All), CUU(2000) and TSGSU against 10,000 unlabeled samples on high dimensional datasets, where the sizes of similar pairs are fixed at 5000. The non-linear model is built by dot product kernel. (The results of SUDH and SUSL are missing since they are out of memory)

$$\begin{aligned}
|A_i(x)| &\leq |a_i^j| (|\zeta_i(x)| + |\xi_i(x)|) \\
&= |a_i^j| (|\delta'_1| \cdot \left(\frac{\pi_S}{2\pi_+ - 1} l'(f(\tilde{x}_i^s), 1)k(\tilde{x}_i^s, x) - \frac{\pi_-}{2\pi_+ - 1} l'(f(x_i^u), 1)k(x_i^u, x) \right) \\
&\quad + \delta'_2 \cdot \left(\frac{\pi_+}{2\pi_+ - 1} l'(f(x_i^u), -1)k(x_i^u, x) - \frac{\pi_S}{2\pi_+ - 1} l'(f(\tilde{x}_i^s), -1)k(\tilde{x}_i^s, x) \right) | \\
&\quad + |\delta'_1| \cdot \left(\frac{\pi_S}{2\pi_+ - 1} l'(f(\tilde{x}_i^s), 1)\phi_\omega(\tilde{x}_i^s)\phi_\omega(x) - \frac{\pi_-}{2\pi_+ - 1} l'(f(x_i^u), 1)\phi_\omega(x_i^u)\phi_\omega(x) \right) \\
&\quad + \delta'_2 \cdot \left(\frac{\pi_+}{2\pi_+ - 1} l'(f(x_i^u), -1)\phi_\omega(x_i^u)\phi_\omega(x) - \frac{\pi_S}{2\pi_+ - 1} l'(f(\tilde{x}_i^s), -1)\phi_\omega(\tilde{x}_i^s)\phi_\omega(x) \right) | \\
&\leq |a_i^j| (|\delta'_1| \cdot \left(\frac{\pi_S}{2\pi_+ - 1} l'(f(\tilde{x}_i^s), 1)k(\tilde{x}_i^s, x) \right) + |\delta'_1| \cdot \left(\frac{\pi_-}{2\pi_+ - 1} l'(f(x_i^u), 1)k(x_i^u, x) \right) | \\
&\quad + |\delta'_2| \cdot \left(\frac{\pi_+}{2\pi_+ - 1} l'(f(x_i^u), -1)k(x_i^u, x) \right) + |\delta'_2| \cdot \left(\frac{\pi_S}{2\pi_+ - 1} l'(f(\tilde{x}_i^s), -1)k(\tilde{x}_i^s, x) \right) | \\
&\quad + |\delta'_1| \cdot \left(\frac{\pi_S}{2\pi_+ - 1} l'(f(\tilde{x}_i^s), 1)\phi_\omega(\tilde{x}_i^s)\phi_\omega(x) \right) + |\delta'_1| \cdot \left(\frac{\pi_-}{2\pi_+ - 1} l'(f(x_i^u), 1)\phi_\omega(x_i^u)\phi_\omega(x) \right) | \\
&\quad + |\delta'_2| \cdot \left(\frac{\pi_+}{2\pi_+ - 1} l'(f(x_i^u), -1)\phi_\omega(x_i^u)\phi_\omega(x) \right) + |\delta'_2| \cdot \left(\frac{\pi_S}{2\pi_+ - 1} l'(f(\tilde{x}_i^s), -1)\phi_\omega(\tilde{x}_i^s)\phi_\omega(x) \right) |) \\
&\leq |a_i^j| \left(M_\delta \frac{\pi_S}{2\pi_+ - 1} M\sqrt{\kappa} + M_\delta \frac{\pi_-}{2\pi_+ - 1} M\sqrt{\kappa} + M_\delta \frac{\pi_+}{2\pi_+ - 1} M\sqrt{\kappa} + M_\delta \frac{\pi_S}{2\pi_+ - 1} M\sqrt{\kappa} \right) \\
&\quad + |a_i^j| \left(M_\delta \frac{\pi_S}{2\pi_+ - 1} M\sqrt{\phi} + M_\delta \frac{\pi_-}{2\pi_+ - 1} M\sqrt{\phi} + M_\delta \frac{\pi_+}{2\pi_+ - 1} M\sqrt{\phi} + M_\delta \frac{\pi_S}{2\pi_+ - 1} M\sqrt{\phi} \right) \\
&= \frac{2\pi_S + 1}{2\pi_+ - 1} M_\delta M (\sqrt{\kappa} + \sqrt{\phi}) |a_i^j|.
\end{aligned} \tag{28}$$

Then, based on the definition of a_t^j and a fix stepsize $\eta_t = \bar{\eta}$, we have $a_t^j \leq \bar{\eta}$. In addition, for any i we have $|a_t^j| \leq \bar{\eta} \prod_{j=i+1}^t (1 - \bar{\eta}\lambda) \leq \bar{\eta}$ if $0 < \bar{\eta}\lambda \leq 1$. Therefore, we have $\sum_{i=1}^t |a_t^j|^2 \leq t\bar{\eta}^2$. Then, for the t -th iteration, we have

$$\mathbb{E}_{\tilde{x}^s, x^u, \omega} [|f_{t+1}(x) - h_{t+1}(x)|^2] \leq \left(\frac{2\pi_S + 1}{2\pi_+ - 1} \right)^2 M_\delta^2 M^2 (\sqrt{\kappa} + \sqrt{\phi})^2 t \bar{\eta}^2. \quad \text{Taking the stepsize}$$

$\bar{\eta} = \frac{\theta}{\sqrt{T}}$, we have

$$\begin{aligned}
B_{1,t+1}^2 &:= \left(\frac{2\pi_S + 1}{2\pi_+ - 1} \right)^2 M_\delta^2 M^2 (\sqrt{\kappa} + \sqrt{\phi})^2 t \frac{\theta^2}{T} \\
&\leq \left(\frac{2\pi_S + 1}{2\pi_+ - 1} \right)^2 M_\delta^2 M^2 (\sqrt{\kappa} + \sqrt{\phi})^2 T \frac{\theta^2}{T} \\
&\leq \left(\frac{2\pi_S + 1}{2\pi_+ - 1} \right)^2 M_\delta^2 M^2 (\sqrt{\kappa} + \sqrt{\phi})^2 \theta^2
\end{aligned}$$

Thus, for the T th iteration, we have

$$\mathbb{E}_{\tilde{x}^s, x^u, \omega} [|f_{T+1}(x) - h_{T+1}(x)|^2] \leq B_{1,T+1}^2,$$

That completes the proof. $\square$

7.2 Proof of Theorem 1

In this subsection, we give the detailed proof of Theorem 1

Proof For convenience, we denote the following three different gradient terms,

$$\begin{aligned}
 g_t &= \xi_t + \lambda h_t \\
 &= \delta'_1 \cdot \left(\frac{\pi_S}{2\pi_+ - 1} l'(f_t(\tilde{x}_t^s), 1) k(\tilde{x}_t^s, \cdot) - \frac{\pi_-}{2\pi_+ - 1} l'(f_t(x_t^\mu), 1) k(x_t^\mu, \cdot) \right) \\
 &\quad + \delta'_2 \cdot \left(\frac{\pi_+}{2\pi_+ - 1} l'(f_t(x_t^\mu), -1) k(x_t^\mu, \cdot) - \frac{\pi_S}{2\pi_+ - 1} l'(f_t(\tilde{x}_t^s), -1) k(\tilde{x}_t^s, \cdot) \right) + \lambda h_t, \\
 \hat{g}_t &= \hat{\xi}_t + \lambda h_t \\
 &= \delta'_1 \cdot \left(\frac{\pi_S}{2\pi_+ - 1} l'(h_t(\tilde{x}_t^s), 1) k(\tilde{x}_t^s, \cdot) - \frac{\pi_-}{2\pi_+ - 1} l'(h_t(x_t^\mu), 1) k(x_t^\mu, \cdot) \right) \\
 &\quad + \delta'_2 \cdot \left(\frac{\pi_+}{2\pi_+ - 1} l'(h_t(x_t^\mu), -1) k(x_t^\mu, \cdot) - \frac{\pi_S}{2\pi_+ - 1} l'(h_t(\tilde{x}_t^s), -1) k(\tilde{x}_t^s, \cdot) \right) + \lambda h_t, \\
 \nabla J(h_t) &= \mathbb{E}_{\tilde{x}_t^s} \mathbb{E}_{x_t^\mu} [\hat{g}_t] \\
 &= \mathbb{E}_{\tilde{x}_t^s} \mathbb{E}_{x_t^\mu} \left[\delta'_1 \cdot \left(\frac{\pi_S}{2\pi_+ - 1} l'(h_t(\tilde{x}_t^s), 1) k(\tilde{x}_t^s, \cdot) - \frac{\pi_-}{2\pi_+ - 1} l'(h_t(x_t^\mu), 1) k(x_t^\mu, \cdot) \right) \right. \\
 &\quad \left. + \delta'_2 \cdot \left(\frac{\pi_+}{2\pi_+ - 1} l'(h_t(x_t^\mu), -1) k(x_t^\mu, \cdot) - \frac{\pi_S}{2\pi_+ - 1} l'(h_t(\tilde{x}_t^s), -1) k(\tilde{x}_t^s, \cdot) \right) \right] + \lambda h_t.
 \end{aligned}$$

From our previous definition, we have $h_{t+1} = h_t - \eta_t g_t$, $\forall t \geq 1$. For $t = 1, \dots, T$, we have

$$\begin{aligned}
 J(h_{t+1}) &\leq J(h_t) + \langle \nabla J(h_t), h_{t+1} - h_t \rangle + \frac{L_j}{2} \|h_{t+1} - h_t\|_{\mathcal{H}}^2 \\
 &= J(h_t) - \eta_t \langle \nabla J(h_t), g_t \rangle + \frac{L_j \eta_t^2}{2} \|g_t\|_{\mathcal{H}}^2 \\
 &= J(h_t) - \eta_t \langle \nabla J(h_t), g_t - \hat{g}_t + \hat{g}_t - \nabla J(h_t) + \nabla J(h_t) \rangle + \frac{L_j \eta_t^2}{2} \|g_t\|_{\mathcal{H}}^2 \\
 &= J(h_t) - \eta_t \|\nabla J(h_t)\|_{\mathcal{H}}^2 + \eta_t \langle \nabla J(h_t), \hat{g}_t - g_t \rangle + \eta_t \langle \nabla J(h_t), \nabla J(h_t) - \hat{g}_t \rangle + \frac{L_j \eta_t^2}{2} \|g_t\|_{\mathcal{H}}^2.
 \end{aligned}$$

Taking expectation on the both side, we can obtain

$$\begin{aligned}
& \eta_t \mathbb{E}[\|\nabla J(h_t)\|_{\mathcal{H}}^2] \\
& \leq \mathbb{E}[J(h_t)] - \mathbb{E}[J(h_{t+1})] + \eta_t \mathbb{E}[\langle \nabla J(h_t), \hat{g}_t - g_t \rangle] + \eta_t \mathbb{E}[\langle \nabla J(h_t), \nabla J(h_t) - \hat{g}_t \rangle] + \frac{L_j \eta_t^2}{2} \mathbb{E}[\|g_t\|_{\mathcal{H}}^2] \\
& \leq \mathbb{E}[J(h_t)] - \mathbb{E}[J(h_{t+1})] + \eta_t \mathbb{E}[\langle \nabla J(h_t), \hat{g}_t - g_t \rangle] + \eta_t \mathbb{E}[\langle \nabla J(h_t), \nabla J(h_t) - \hat{g}_t \rangle] + \frac{L_j \eta_t^2}{2} \mathbb{E}[\|g_t\|_{\mathcal{H}}^2],
\end{aligned}$$

where J^* denotes the optimal value of $J(h)$ and $\mathbb{E}[\cdot] = \mathbb{E}_{\bar{x}_t^i, \bar{x}_t^u, \omega_t}[\cdot]$. Let us denote $\mathcal{H}_t = \sqrt{\|\nabla J(h_t)\|_{\mathcal{H}}^2}$, $\mathcal{M}_t = \|g_t\|_{\mathcal{H}}^2$, $\mathcal{N}_t = \langle \nabla J(h_t), \nabla J(h_t) - \hat{g}_t \rangle$ and $\mathcal{R}_t = \langle \nabla J(h_t), \hat{g}_t - g_t \rangle$. For the fixed stepsize $\eta_t = \bar{\eta} = \frac{\theta}{\sqrt{T}}$, based on Lemma 2, we have

$$\begin{aligned}
& \bar{\eta} \mathbb{E}[\|\nabla J(h_t)\|_{\mathcal{H}}^2] \\
& \leq \mathbb{E}[J(h_t)] - \mathbb{E}[J(h_{t+1})] + \bar{\eta}(1 + \lambda) \left(\frac{2\pi_S + 1}{2\pi_+ - 1} \right)^2 M_\delta^2 M L \kappa B_{1,t} + 2\bar{\eta}^2 L_j \left(\frac{\pi_S + 1}{2\pi_+ - 1} \right)^2 M_\delta^2 M^2 \kappa \\
& \leq \mathbb{E}[J(h_t)] - \mathbb{E}[J(h_{t+1})] + (1 + \lambda) \left(\frac{2\pi_S + 1}{2\pi_+ - 1} \right)^3 M_\delta^3 M^2 L \kappa (\sqrt{\kappa} + \sqrt{\phi}) \theta \bar{\eta} + 2\bar{\eta}^2 L_j^2 \left(\frac{\pi_S + 1}{2\pi_+ - 1} \right)^2 M_\delta^2 M^2 \kappa \\
& \hspace{15em} (29)
\end{aligned}$$

. Summing both sides of the inequality for $t \in \{1, \dots, T\}$ and recall the Assumption, we have

$$\begin{aligned}
& \bar{\eta} \mathbb{E} \left[\sum_{t=1}^T \|\nabla J(h_t)\|_{\mathcal{H}}^2 \right] \\
& \leq J(h_1) - J_{\inf} + (1 + \lambda) \left(\frac{2\pi_S + 1}{2\pi_+ - 1} \right)^3 M_\delta^3 M^2 L \kappa (\sqrt{\kappa} + \sqrt{\phi}) \theta \bar{\eta} T + 2L_j \bar{\eta}^2 \left(\frac{\pi_S + 1}{2\pi_+ - 1} \right)^2 M_\delta^2 M^2 \kappa T
\end{aligned}$$

Rearranging the above inequality and dividing by T , we have

$$\begin{aligned}
& \mathbb{E} \left[\frac{1}{T} \sum_{t=1}^T \|\nabla J(h_t)\|_{\mathcal{H}}^2 \right] \\
& \leq \frac{J(h_1) - J_{\inf}}{\bar{\eta} T} + (1 + \lambda) \left(\frac{2\pi_S + 1}{2\pi_+ - 1} \right)^3 M_\delta^3 M^2 L \kappa (\sqrt{\kappa} + \sqrt{\phi}) \theta \bar{\eta} + 2L_j \bar{\eta} \left(\frac{\pi_S + 1}{2\pi_+ - 1} \right)^2 M_\delta^2 M^2 \kappa \\
& = \frac{J(h_1) - J_{\inf}}{\theta \sqrt{T}} + (1 + \lambda) \left(\frac{2\pi_S + 1}{2\pi_+ - 1} \right)^3 M_\delta^3 M^2 L \kappa (\sqrt{\kappa} + \sqrt{\phi}) \frac{\theta^2}{\sqrt{T}} + 2L_j \frac{\theta}{\sqrt{T}} \left(\frac{\pi_S + 1}{2\pi_+ - 1} \right)^2 M_\delta^2 M^2 \kappa
\end{aligned}$$

That completes the proof. $\square$

7.3 Lemma 2

In this subsection, we give Lemma 2 and its detailed proof.

Lemma 2 Let us denote $\mathcal{H}_t = \sqrt{\|\nabla J(h_t)\|_{\mathcal{H}}^2}$, $\mathcal{M}_t = \|g_t\|_{\mathcal{H}}^2$, $\mathcal{N}_t = \langle \nabla J(h_t), \nabla J(h_t) - \hat{g}_t \rangle$ and $\mathcal{R}_t = \langle \nabla J(h_t), \hat{g}_t - g_t \rangle$. $\mathcal{H}_t$, $\mathcal{M}_t$, $\mathcal{N}_t$ and $\mathcal{R}_t$ are bounded as follows,

$$\mathcal{M}_t \leq 4 \left(\frac{\pi_S + 1}{2\pi_+ - 1} \right)^2 M_\delta^2 M^2 \kappa, \quad (30)$$

$$\mathbb{E}^2[\mathcal{H}_t] \leq (1 + \lambda)^2 \left(\frac{\pi_S + 1}{2\pi_+ - 1} \right)^2 M_\delta^2 M^2 \kappa, \quad (31)$$

$$\mathbb{E}[\mathcal{N}_t] = 0, \quad (32)$$

$$\mathbb{E}[\mathcal{R}_t] \leq (1 + \lambda) \left(\frac{2\pi_S + 1}{2\pi_+ - 1} \right)^2 M_\delta^2 M L \kappa B_{1,t}. \quad (33)$$

Proof (1). First, we give the bound of $\mathcal{M}_t$.

$$\mathcal{M}_t = \|g_t\|_{\mathcal{H}}^2 = \|\xi_t + \lambda h_t\|_{\mathcal{H}}^2 \leq (\|\xi_t\|_{\mathcal{H}} + \lambda \|h_t\|_{\mathcal{H}})^2 \quad (34)$$

If we can give the bound of $\|\xi_t\|_{\mathcal{H}}$ and $\|h_t\|_{\mathcal{H}}$, then we can obtain the bound of $\mathcal{M}_t$.

$$\begin{aligned} \|\xi_t\|_{\mathcal{H}} &= \|\delta'_1 \cdot \left(\frac{\pi_S}{2\pi_+ - 1} l'(f(\tilde{x}_t^s), 1)k(\tilde{x}_t^s, \cdot) - \frac{\pi_-}{2\pi_+ - 1} l'(f(x_t^u), 1)k(x_t^u, \cdot) \right) \\ &\quad + \delta'_2 \cdot \left(\frac{\pi_+}{2\pi_+ - 1} l'(f(x_t^u), -1)k(x_t^u, \cdot) - \frac{\pi_S}{2\pi_+ - 1} l'(f(\tilde{x}_t^s), -1)k(\tilde{x}_t^s, \cdot) \right)\|_{\mathcal{H}} \\ &\leq \left\| \delta'_1 \cdot \frac{\pi_S}{2\pi_+ - 1} l'(f(\tilde{x}_t^s), 1)k(\tilde{x}_t^s, \cdot) \right\|_{\mathcal{H}} + \left\| \delta'_1 \cdot \frac{\pi_-}{2\pi_+ - 1} l'(f(x_t^u), 1)k(x_t^u, \cdot) \right\|_{\mathcal{H}} \\ &\quad + \left\| \delta'_2 \cdot \frac{\pi_+}{2\pi_+ - 1} l'(f(x_t^u), -1)k(x_t^u, \cdot) \right\|_{\mathcal{H}} + \left\| \delta'_2 \cdot \frac{\pi_S}{2\pi_+ - 1} l'(f(\tilde{x}_t^s), -1)k(\tilde{x}_t^s, \cdot) \right\|_{\mathcal{H}} \\ &\leq M_\delta \frac{\pi_S}{2\pi_+ - 1} M \kappa^{1/2} + M_\delta \frac{\pi_-}{2\pi_+ - 1} M \kappa^{1/2} + M_\delta \frac{\pi_+}{2\pi_+ - 1} M \kappa^{1/2} + M_\delta \frac{\pi_S}{2\pi_+ - 1} M \kappa^{1/2} \\ &= \frac{2\pi_S + 1}{2\pi_+ - 1} M_\delta M \kappa^{1/2} \end{aligned}$$

For $t = 1$, according to the definition of h_t , we have $h_1 = 0$ and $\|h_1\|_{\mathcal{H}} = 0$. In addition, assume that $\|h_t\|_{\mathcal{H}} \leq \frac{2\pi_S + 1}{2\pi_+ - 1} M_\delta M \kappa^{1/2}$ for any $t = 1, \dots, T - 1$, we have

$$\begin{aligned}
\|h_{t+1}\|_{\mathcal{H}} &= \|(1 - \eta_t)h_t - \eta_t \xi_t(\cdot)\|_{\mathcal{H}} \\
&\leq (1 - \eta_t)\|h_t\|_{\mathcal{H}} + \eta_t\|\xi_t(\cdot)\|_{\mathcal{H}} \\
&\leq (1 - \eta_t)\frac{2\pi_S + 1}{2\pi_+ - 1}M_\delta M\kappa^{1/2} + \eta_t\frac{2\pi_S + 1}{2\pi_+ - 1}M_\delta M\kappa^{1/2} \\
&\leq \frac{2\pi_S + 1}{2\pi_+ - 1}M_\delta M\kappa^{1/2}
\end{aligned}$$

Thus, based on the above two inequalities, we have

$$\mathcal{M}_t \leq 4 \left(\frac{2\pi_S + 1}{2\pi_+ - 1} \right)^2 M_\delta^2 M^2 \kappa$$

(2). Then, we prove the bound of $\mathcal{H}_t$.

$$\begin{aligned}
\mathbb{E}^2[\mathcal{H}_t] &= \mathbb{E}_{\tilde{x}_t^s, x_t^\mu, \omega_t}^2[\mathcal{H}_t] \\
&= \mathbb{E}_{\tilde{x}_t^s, x_t^\mu, \omega_t}[\|\nabla J(h_t)\|_{\mathcal{H}}^2] \\
&= \left\| \mathbb{E}_{\tilde{x}_t^s, x_t^\mu, \omega_t}[\hat{\xi}_t] + \lambda h_t \right\|_{\mathcal{H}}^2 \\
&\leq \left(\left\| \mathbb{E}_{\tilde{x}_t^s, x_t^\mu, \omega_t}[\hat{\xi}_t] \right\|_{\mathcal{H}} + \lambda \|h_t\|_{\mathcal{H}} \right)^2
\end{aligned}$$

According to the above results, we have $\left\| \mathbb{E}_{\tilde{x}_t^s, x_t^\mu, \omega_t}[\hat{\xi}_t] \right\|_{\mathcal{H}} \leq \frac{2\pi_S + 1}{2\pi_+ - 1}M_\delta M\kappa^{1/2}$ and $\|h_t\|_{\mathcal{H}} \leq \frac{2\pi_S + 1}{2\pi_+ - 1}M_\delta M\kappa^{1/2}$. Therefore, we have

$$\mathbb{E}^2[\mathcal{H}_t] \leq (1 + \lambda)^2 \left(\frac{2\pi_S + 1}{2\pi_+ - 1} \right)^2 M_\delta^2 M^2 \kappa \quad (35)$$

(3). Here, we give the bound of $\mathcal{N}_t$.

$$\begin{aligned}
\mathbb{E}[\mathcal{N}_t] &= \mathbb{E}_{\tilde{x}_t^s, x_t^\mu, \omega_t}[\mathcal{N}_t] \\
&= \mathbb{E}_{\tilde{x}_t^s, x_t^\mu, \omega_t} \left[\mathbb{E}_{\tilde{x}_t^s, x_t^\mu} [\langle \nabla J(h_t), \nabla J(h_t) - \hat{g}_t \rangle | \tilde{x}_{t-1}^s, x_{t-1}^\mu, \omega_t] \right] \\
&= \mathbb{E}_{\tilde{x}_t^s, x_t^\mu, \omega_t} \left[\langle \nabla J(h_t), \mathbb{E}_{\tilde{x}_t^s, x_t^\mu} [\nabla J(h_t) - \hat{g}_t] \rangle | \tilde{x}_{t-1}^s, x_{t-1}^\mu, \omega_t \right] \\
&= 0
\end{aligned} \quad (36)$$

(4). Finally, we bound $\mathcal{R}_t$.

$$\begin{aligned}
\mathbb{E}[\mathcal{R}_t] &= \mathbb{E}_{\tilde{x}_t^s, x_t^u, \omega_t}[\mathcal{R}_t] \\
&= \mathbb{E}_{\tilde{x}_t^s, x_t^u, \omega_t}[\langle \nabla J(h_t), g_t - \hat{g}_t \rangle] \\
&= \mathbb{E}_{\tilde{x}_t^s, x_t^u, \omega_t}[\langle \nabla J(h_t), \delta'_1 \frac{\pi_S}{2\pi_+ - 1} k(\tilde{x}_t^s, \cdot)(l'(f_t(\tilde{x}_t^s), 1) - l'(h_t(\tilde{x}_t^s), 1)) \\
&\quad - \delta'_1 \frac{\pi_-}{2\pi_+ - 1} k(x_t^u, \cdot)(l'(f_t(x_t^u), 1) - l'(h_t(x_t^u), 1)) \rangle] \\
&\quad + \mathbb{E}_{\tilde{x}_t^s, x_t^u, \omega_t}[\langle \nabla J(h_t), \delta'_2 \frac{\pi_+}{2\pi_+ - 1} k(x_t^u, \cdot)(l'(f_t(x_t^u), -1) - l'(h_t(x_t^u), -1)) \\
&\quad - \delta'_1 \frac{\pi_S}{2\pi_+ - 1} k(\tilde{x}_t^s, \cdot)(l'(f_t(\tilde{x}_t^s), -1) - l'(h_t(\tilde{x}_t^s), -1)) \rangle] \\
&\leq \mathbb{E}_{\tilde{x}_t^s, x_t^u, \omega_t}[\|\nabla J(h_t)\|_{\mathcal{H}} \cdot \|\delta'_1 \frac{\pi_S}{2\pi_+ - 1} k(\tilde{x}_t^s, \cdot)(l'(f_t(\tilde{x}_t^s), 1) - l'(h_t(\tilde{x}_t^s), 1)) \\
&\quad - \delta'_1 \frac{\pi_-}{2\pi_+ - 1} k(x_t^u, \cdot)(l'(f_t(x_t^u), 1) - l'(h_t(x_t^u), 1))\|] \\
&\quad + \mathbb{E}_{\tilde{x}_t^s, x_t^u, \omega_t}[\|\nabla J(h_t)\|_{\mathcal{H}} \cdot \|\delta'_2 \frac{\pi_+}{2\pi_+ - 1} k(x_t^u, \cdot)(l'(f_t(x_t^u), -1) - l'(h_t(x_t^u), -1)) \\
&\quad - \delta'_1 \frac{\pi_S}{2\pi_+ - 1} k(\tilde{x}_t^s, \cdot)(l'(f_t(\tilde{x}_t^s), -1) - l'(h_t(\tilde{x}_t^s), -1))\|] \\
&\leq \frac{\pi_S}{2\pi_+ - 1} \kappa^{1/2} M_\delta L \mathbb{E}_{\tilde{x}_t^s, x_t^u, \omega_t} \left[\|\nabla J(h_t)\|_{\mathcal{H}} \cdot \left| (f_t(\tilde{x}_t^s) - h_t(\tilde{x}_t^s)) \right| \right] \\
&\quad + \frac{\pi_-}{2\pi_+ - 1} \kappa^{1/2} M_\delta L \mathbb{E}_{\tilde{x}_t^s, x_t^u, \omega_t} \left[\|\nabla J(h_t)\|_{\mathcal{H}} \cdot |f_t(x_t^u) - h_t(x_t^u)| \right] \\
&\quad + \frac{\pi_+}{2\pi_+ - 1} \kappa^{1/2} M_\delta L \mathbb{E}_{\tilde{x}_t^s, x_t^u, \omega_t} \left[\|\nabla J(h_t)\|_{\mathcal{H}} \cdot |f_t(x_t^u) - h_t(x_t^u)| \right] \\
&\quad + \frac{\pi_S}{2\pi_+ - 1} \kappa^{1/2} M_\delta L \mathbb{E}_{\tilde{x}_t^s, x_t^u, \omega_t} \left[\|\nabla J(h_t)\|_{\mathcal{H}} \cdot |f_t(\tilde{x}_t^s) - h_t(\tilde{x}_t^s)| \right] \\
&\leq \frac{\pi_S}{2\pi_+ - 1} \kappa^{1/2} M_\delta L \sqrt{\mathbb{E}_{\tilde{x}_t^s, x_t^u, \omega_t} [\|\nabla J(h_t)\|_{\mathcal{H}}^2]} \cdot \sqrt{\mathbb{E}_{\tilde{x}_t^s, x_t^u, \omega_t} \left[\left| (f_t(\tilde{x}_t^s) - h_t(\tilde{x}_t^s)) \right|^2 \right]} \\
&\quad + \frac{\pi_-}{2\pi_+ - 1} \kappa^{1/2} M_\delta L \sqrt{\mathbb{E}_{\tilde{x}_t^s, x_t^u, \omega_t} [\|\nabla J(h_t)\|_{\mathcal{H}}^2]} \cdot \sqrt{\mathbb{E}_{\tilde{x}_t^s, x_t^u, \omega_t} \left[|f_t(x_t^u) - h_t(x_t^u)|^2 \right]} \\
&\quad + \frac{\pi_+}{2\pi_+ - 1} \kappa^{1/2} M_\delta L \sqrt{\mathbb{E}_{\tilde{x}_t^s, x_t^u, \omega_t} [\|\nabla J(h_t)\|_{\mathcal{H}}^2]} \cdot \sqrt{\mathbb{E}_{\tilde{x}_t^s, x_t^u, \omega_t} \left[|f_t(x_t^u) - h_t(x_t^u)|^2 \right]} \\
&\quad + \frac{\pi_S}{2\pi_+ - 1} \kappa^{1/2} M_\delta L \sqrt{\mathbb{E}_{\tilde{x}_t^s, x_t^u, \omega_t} [\|\nabla J(h_t)\|_{\mathcal{H}}^2]} \cdot \sqrt{\mathbb{E}_{\tilde{x}_t^s, x_t^u, \omega_t} \left[|f_t(\tilde{x}_t^s) - h_t(\tilde{x}_t^s)|^2 \right]} \\
&\leq \frac{\pi_S}{2\pi_+ - 1} \kappa^{1/2} M_\delta L B_{1,t} \sqrt{\mathbb{E}_{\tilde{x}_t^s, x_t^u, \omega_t} [\mathcal{H}_t]} + \frac{\pi_-}{2\pi_+ - 1} \kappa^{1/2} M_\delta L B_{1,t} \sqrt{\mathbb{E}_{\tilde{x}_t^s, x_t^u, \omega_t} [\mathcal{H}_t]} \\
&\quad + \frac{\pi_+}{2\pi_+ - 1} \kappa^{1/2} M_\delta L B_{1,t} \sqrt{\mathbb{E}_{\tilde{x}_t^s, x_t^u, \omega_t} [\mathcal{H}_t]} + \frac{\pi_S}{2\pi_+ - 1} \kappa^{1/2} M_\delta L B_{1,t} \sqrt{\mathbb{E}_{\tilde{x}_t^s, x_t^u, \omega_t} [\mathcal{H}_t]} \\
&\leq (1 + \lambda) \left(\frac{2\pi_S + 1}{2\pi_+ - 1} \right)^2 M_\delta^2 M L \kappa B_{1,t}
\end{aligned} \tag{37}$$

That completes the proof. $\square$

8 Conclusion

In this paper, we propose a novel scalable kernel SU classification algorithm, TSGSU. To update the model function in a stochastic optimization framework, we randomly sample an instance from the similar pairs dataset, another instance from the unlabeled dataset, and their random features to calculate the triply stochastic gradient. Then the model can be updated by using this approximated gradient. Even though the SU problem is non-convex, theoretically, we prove that TSGSU has a convergence rate of $O(1/\sqrt{T})$. The experimental results on various datasets also demonstrate the superiority of the proposed TSGSU.

Acknowledgements This work was supported by the Project Funded by the Priority Academic Program Development (PAPD) of Jiangsu Higher Education Institutions, Six talent peaks project (No. XYDXX-042) and the 333 Project (No. BRA2017455) in Jiangsu Province and the National Natural Science Foundation of China (No:61573191).

References

- Bao, H., Niu, G., & Sugiyama, M. (2018). Classification from pairwise similarity and unlabeled data. In: *International Conference on Machine Learning*, (pp. 461–470).
- Calandriello, D., Niu, G., & Sugiyama, M. (2014). Semi-supervised information-maximization clustering. *Neural Networks*, 57, 103–111.
- Chapelle, O., Scholkopf, B., & Zien, A. (2009). Semi-supervised learning (chapelle, o. et al., eds.; 2006) [book reviews]. *IEEE Transactions on Neural Networks*, 20(3), 542–542.
- Dai, B., Xe, B., He, N., Liang, Y., Raj, A., Balcan, M. F., & Song, L. (2014). Scalable kernel methods via doubly stochastic gradients. In: *Advances in Neural Information Processing Systems*, (pp. 3041–3049).
- Drineas, P., & Mahoney, M. W. (2005). On the nystrom method for approximating a gram matrix for improved kernel-based learning. *Journal of Machine Learning Research*, 6(Dec), 2153–2175.
- du Plessis MC, Niu G, & Sugiyama M (2014). Analysis of learning from positive and unlabeled data. In: *Advances in neural information processing systems*, (pp. 703–711).
- du Plessis, M. C, Niu, G., & Sugiyama, M. (2015a). Class-prior estimation for learning from positive and unlabeled data. In: *ACML*, (pp. 221–236).
- du Plessis, M. C., Niu, G., & Sugiyama, M. (2015b). Convex formulation for learning from positive and unlabeled data. In: *International Conference on Machine Learning*, (pp. 1386–1394).
- Fine, S., & Scheinberg, K. (2001). Efficient svm training using low-rank kernel representations. *Journal of Machine Learning Research*, 2(Dec), 243–264.
- Geng, X., Gu, B., Li, X., Shi, W., Zheng, G., & Huang, H. (2019). Scalable semi-supervised svm via triply stochastic gradients. In: *28th International Joint Conference on Artificial Intelligence*.
- Gu, B., Huo, Z., & Huang, H. (2016). Asynchronous stochastic block coordinate descent with variance reduction. arXiv preprint arXiv:1610.09447.
- Gu, B., Xin, M., Huo, Z., & Huang, H. (2018a). Asynchronous doubly stochastic sparse kernel learning. In: *Thirty-Second AAAI Conference on Artificial Intelligence*.
- Gu, B., Xin, M., Huo, Z., & Huang, H. (2018b). Asynchronous doubly stochastic sparse kernel learning. In: *AAAI Conference on Artificial Intelligence*.
- Khan, S. S, & Madden, M. G. (2009). A survey of recent trends in one class classification. In: *Irish conference on artificial intelligence and cognitive science*, (pp. 188–197). Springer, Berlin
- Kiryo, R., Niu, G., du Plessis M. C., & Sugiyama, M. (2017). Positive-unlabeled learning with non-negative risk estimator. In: *Advances in Neural Information Processing Systems*, (pp. 1675–1685).
- Le, Q., Sarlós, T., & Smola, A. (2013). Fastfood-computing hilbert space expansions in loglinear time. In: *International Conference on Machine Learning*, (pp. 244–252).
- Li, X., Gu, B., Ao, S., Wang, H., & Ling, C. X. (2017). Triply stochastic gradients on multiple kernel learning. Conference on Uncertainty in Artificial Intelligence.
- Lu, N., Niu, G., Menon, A. K., & Sugiyama, M. (2019). On the minimal supervision for training any binary classifier from only unlabeled data. In *Proceedings of the 7th International Conference on Learning Representations (ICLR'19)*, 18 pages, New Orleans, Louisiana, USA, May 6–9,.

- Lu, N., Zhang, T., Niu, G., & Sugiyama, M., (2020). Mitigating overfitting in supervised classification from two unlabeled datasets: A consistent risk correction approach. In: *International Conference on Artificial Intelligence and Statistics*, (pp. 1115–1125).
- Munkhoeva, M., Kapushev, Y., Burnaev, E., & Oseledets, I. (2018). Quadrature-based features for kernel approximation. arXiv preprint arXiv:1802.03832.
- Pham, N., & Pagh, R. (2013). Fast and scalable polynomial kernels via explicit feature maps. In: *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, (pp. 239–247).
- Rahimi, A., & Recht, B. (2008). Random features for large-scale kernel machines. In: *Advances in neural information processing systems*, (pp. 1177–1184).
- Rahimi, A., & Recht, B. (2009). Weighted sums of random kitchen sinks: Replacing minimization with randomization in learning. In: *Advances in neural information processing systems*, (pp. 1313–1320).
- Rakhlin, A., Shamir, O., & Sridharan, K. (2012). Making gradient descent optimal for strongly convex stochastic optimization. In: *International Conference on Machine Learning*, (pp. 1571–1578).
- Ramaswamy, H., Scott, C., & Tewari, A. (2016). Mixture proportion estimation via kernel embeddings of distributions. In: *International conference on machine learning*, (pp. 2052–2060). PMLR.
- Sakai, T., du Plessis, M. C., Niu, G., & Sugiyama, M. (2017). Semi-supervised classification based on classification from positive and unlabeled data. In: *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, (pp. 2998–3006). JMLR. org.
- Sakai, T., Niu, G., & Sugiyama, M. (2018). Semi-supervised auc optimization based on positive-unlabeled learning. *Machine Learning*, 107(4), 767–794.
- Schölkopf, B., Platt, J. C., Shawe-Taylor, J., Smola, A. J., & Williamson, R. C. (2001). Estimating the support of a high-dimensional distribution. *Neural Computation*, 13(7), 1443–1471.
- Scott, C. (2015). A rate of convergence for mixture proportion estimation, with application to learning from noisy labels. In: *Artificial Intelligence and Statistics*, (pp. 838–846). PMLR.
- Shi, W., Gu, B., Li, X., Geng, X., & Huang, H. (2019). Quadruply stochastic gradients for large scale nonlinear semi-supervised auc optimization. In: *28th International Joint Conference on Artificial Intelligence*.
- Shi, W., Gu, B., Li, X., & Huang, H. (2020). Quadruply stochastic gradient method for large scale nonlinear semi-supervised ordinal regression auc optimization. In: *AAAI Conference on Artificial Intelligence*, (pp. 5734–5741).
- Smola, A. J., & Schölkopf, B. (1998). *Learning with kernels*, volume 4. Citeseer.
- Smola, A. J., & Schölkopf, B. (2000). Sparse greedy matrix approximation for machine learning.
- Vedaldi, A., & Zisserman, A. (2012). Efficient additive kernels via explicit feature maps. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(3), 480–492.
- Yang, J., Sindhvani, V., Fan, Q., Avron, H., & Mahoney, M. W. (2014). Random laplace feature maps for semigroup kernels on histograms. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 971–978.
- Yu, F. X. X., Suresh, A. T., Choromanski, K. M., Holtmann-Rice, D. N., & Kumar, S., (2016). Orthogonal random features. In: *Advances in Neural Information Processing Systems*, (pp. 1975–1983).
- Yu, S., Gu, B., Ning, K., Chen, H., Pei, J., & Huang, H., (2019). Tackle balancing constraint for incremental semi-supervised support vector learning. In: *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*.

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Authors and Affiliations

Wanli Shi^{1,2} · Bin Gu^{1,2,3} · Xiang Li⁴ · Cheng Deng⁵ · Heng Huang^{3,6}

Wanli Shi
wanlishi@nuist.edu.cn

Xiang Li
lixiang2@uwo.ca

Cheng Deng
chdeng.xd@gmail.com

Heng Huang
henghuanghh@gmail.com

- ¹ Nanjing University of Information Science & Technology, Nanjing, China
- ² MBZUAI, Masdar City, United Arab Emirates
- ³ JD Finance America Corporation, Silicon Valley, USA
- ⁴ Computer Science Department, University of Western Ontario, London, ON, Canada
- ⁵ School of Electronic Engineering, Xidian University, Xi'an, China
- ⁶ Department of Electrical & Computer Engineering, University of Pittsburgh, Pittsburgh, USA