Scaling Up Kernel SVM on Limited Resources: A Low-Rank Linearization Approach

Liang Lan, Zhuang Wang, Shandian Zhe, Wei Cheng, Jun Wang, and Kai Zhang

***Abstract*— Kernel support vector machines (SVMs) deliver state-of-the-art results in many real-world nonlinear classification problems, but the computational cost can be quite demanding in order to maintain a large number of support vectors. Linear SVM, on the other hand, is highly scalable to large data but only suited for linearly separable problems. In this paper, we propose a novel approach called low-rank linearized SVM to scale up kernel SVM on limited resources. Our approach transforms a nonlinear SVM to a linear one via an approximate empirical kernel map computed from efficient kernel low-rank decompositions. We theoretically analyze the gap between the solutions of the approximate and optimal rank-*k* kernel map, which in turn provides guidance on the sampling scheme of the Nyström approximation. Furthermore, we extend it to a semisupervised metric learning scenario in which partially labeled samples can be exploited to further improve the quality of the low-rank embedding. Our approach inherits rich representability of kernel SVM and high efficiency of linear SVM. Experimental results demonstrate that our approach is more robust and achieves a better tradeoff between model representability and scalability against state-of-the-art algorithms for large-scale SVMs.**

***Index Terms*— Large-scale learning, low-rank approximation, metric learning, support vector machine (SVM).**

I. Introduction

SUPPORT vector machine (SVM) [1]–[3] is state-of-the-art algorithm for classification and has been widely used in various real-world applications. The use of kernels implicitly maps samples from the input space to the reproducing kernel Hilbert space, which is crucial for solving nonlinear problems. Unfortunately, the need to manipulate the kernel matrix imposes a severe computational bottleneck, making it difficult to scale up to truly large data.

Manuscript received February 3, 2016; revised April 14, 2017 and January 17, 2018; accepted May 9, 2018. Date of publication June 21, 2018; date of current version January 21, 2019. *(Corresponding author: Kai Zhang.)*

L. Lan is with the Department of Computer Science, Hong Kong Baptist University, Hong Kong (e-mail: lanliang@comp.hkbu.edu.hk).

Z. Wang was with Siemens Corporate Research, Princeton, NJ 08540 USA. He is now with Facebook, Menlo Park, CA 94025 USA (e-mail: johnwangzh@fb.com).

S. Zhe is with the School of Computing, University of Utah, Salt Lake City, UT 84103 USA (e-mail: zhe@cs.utah.edu).

W. Cheng is with the NEC Laboratories America, Princeton, NJ 08550 USA (e-mail: wcheng@nec-labs.com).

J. Wang is with the School of Computer Science and Software Engineering, East China Normal University, Shanghai 200062, China (e-mail: wongjun@gmail.com).

K. Zhang is with the Department of Computer and Information Sciences, Temple University, Philadelphia, PA 19122 USA (e-mail: zhang.kai@temple.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TNNLS.2018.2838140

Many optimization schemes have been proposed to scale up kernel SVM training, including decomposition approaches [4], [5], approximation algorithms (e.g., core vector set [6] and incremental learning methods [7]), parallel computing techniques [8], the cutting plane method [9], and divide-and-conquer approach [10]. However, advances in kernel SVM training are still behind the growth of data. The latter leads to the curse of kernelization [11], where the number of support vector increase linearly with the sample size on noisy data. This leads to huge memory and computational costs. For example, training a kernel SVM on an optical character recognition task with 8 million training samples using 512 processors via the parallel computing technique [8] still takes two days. Moreover, the resultant model consists of hundreds of thousands of support vectors, which brings in additional space and time cost for prediction.

Meanwhile, large-scale linear SVM training has also regained tremendous interests recently. Unlike kernel SVM, the model coefficients of a linear SVM can be explicitly computed without the need to keep support vectors. A series of efficient linear SVM training algorithms have been proposed, including the stochastic gradient descent method [12], [13], the cutting plane method [14], and the dual coordinate descent method [15]. These algorithms typically have linear time complexities, and enable efficient training of a linear SVM on gigabytes of data using regular PC's in a matter of minutes, and sometimes even on data sets that cannot fit in memory [16].

Inspired by these advances, researchers started to speed up kernel SVM using efficient linear SVM solvers. For example, Chang *et al.* [17] and Sonnenburg and Franc [18] proposed to precompute the kernel-mapped features and then train a linear SVM on them. However, it only applies to certain types of kernels (e.g., string kernel, low-degree polynomial kernel) in which the dimension of kernel space is not large. Rahimi and Recht [19] proposed randomized Fourier features by approximating the kernel function using the Fourier transform.

Besides data-independent methods [17]–[19], low-rank structures of the kernel matrix are also quite useful in providing informative features [20] and to improve the computational efficiency [21]. A successful example is the Nystöm method [22]–[24], which is a sampling-based method for large kernel eigen-systems. Cortes *et al.* [25] analyzed how the kernel approximation quality affects the performance of kernel machines, shedding light on the quality of low-rank approximation in terms of generalization performance.

Recently, the Nyström method was successfully extended to rectangular matrices, making it possible to obtain approximate singular value decomposition in just $O(m+n)$ time and space for an $m \times n$ matrix, which is more efficient than existing randomized algorithms [26].

Although the Nyström method has drawn considerable interest, there are still open problems on applying it to scale up kernel SVM in both computationally efficient and theoretically relevant manner. For example, most theoretical studies [22], [27] focus on how well the Nyström method can approximate a positive semidefinite (PSD) kernel matrix, but not on how such approximation affects the solution of the SVMs. In [25], although useful theoretical guarantees are provided in terms of the generalization performance, the dimensionality of the kernel map equals the sample size, making the training process computationally as expensive as the original kernel SVM. Furthermore, existing methods typically compute the low-rank approximation regardless of the class labels [21], [22], [25], [27], which can lead to inferior kernel map in training kernel-based classifiers.

In this paper, we seek a general framework on using the Nyström method to generate low-dimensional empirical kernel map, such that kernel SVM training can benefit significantly from recent advances in linear SVM. First, we proposed the use of clustering-based Nyström method to linearize kernel SVM in linear time and space, together with theoretical guarantees on the performance of the resultant predictor. Second, we draw an interesting connection between the our approach and the reduced SVM (RSVM) [28]; and show that our approach provides a more principled solution to the problem raised in [28]. Third, we have developed a highly optimized solver for kernel SVM on data sets that do not fit in memory using selective block minimization (SBM) [29]. Last, we propose an innovative formulation to generalize the Nyström method from purely unsupervised setting to semisupervised scenario, which improves the stability and accuracy of the resultant predictor.

The proposed method applies to any SVM variant and PSD kernels, and the dimension of the approximate kernel map and the model complexity can be controlled conveniently, guaranteeing efficient training and testing. Our approach inherits rich representability of kernel SVM and high efficiency of linear SVM, and can handle arbitrarily large data with limited computing resources via incremental learning techniques [16]. Experimental results have demonstrated that it is a promising direction toward training efficient yet accurate nonlinear SVM models. A conference version can be found in [30].

II. Transforming Nonlinear SVM to Linear

Suppose we are given a training data set $\{\mathbf{X}_r, \mathbf{y}_r\}$, where $\mathbf{X}_r$ is a $n \times d$ matrix and the ith row denotes the ith sample. $\mathbf{y}_r$ is a vector with length n to denote the training labels. Similarly, we have test data $\mathbf{X}_e \in \mathbf{R}^{m\times d}$. Assume we use a PSD kernel function $\kappa(\mathbf{x}_i, \mathbf{x}_j) = \langle \psi(\mathbf{x}_i), \psi(\mathbf{x}_j)\rangle : \mathbf{R}^d \times \mathbf{R}^d \to \mathbf{R}$, where $\psi(\mathbf{x}_i)$ is the mapping function that implicitly maps samples from input space to feature space. Define the kernel matrix on the training and test data in blocks as $\mathbf{K} = \begin{bmatrix} \mathbf{K}_{\text{rr}} & \mathbf{K}_{\text{re}} \\ \mathbf{K}_{\text{er}} & \mathbf{K}_{\text{ee}} \end{bmatrix}$, where $\mathbf{K}_{\text{rr}} \in \mathbf{R}^{n\times n}$ is the kernel matrix defined on $\mathbf{X}_r$, $\mathbf{K}_{\text{ee}} \in \mathbf{R}^{m\times m}$ is defined on $\mathbf{X}_e$, and $\mathbf{K}_{\text{er}} \in \mathbf{R}^{m\times n}$ is defined on $\mathbf{X}_e$ and $\mathbf{X}_r$. Training a kernel SVM is to solve the following optimization problem:

$$\min_{\mathbf{w},\xi,b} \ \frac{1}{2}\|\mathbf{w}\|^2 + C\sum \xi_i \qquad y_i(\mathbf{w}^\top \psi(\mathbf{x}_i) + b) \ge 1 - \xi_i \tag{1}$$

where $C > 0$ is the regularization parameter and b is a bias term. Next, we show how to transform the kernel SVM (1) into a linear one via kernel matrix decomposition.

Proposition 1: Given training data set $\{\mathbf{X}_r, \mathbf{y}_r\}$ and test data $\mathbf{X}_e$, a kernel SVM (1) trained on $\{\mathbf{X}_r, \mathbf{y}_r\}$ and tested on $\mathbf{X}_e$ is equivalent to a linear SVM trained on $\{\mathbf{F}_r, \mathbf{y}_r\}$ and tested on $\mathbf{F}_e$, where

$$\mathbf{K} = \begin{bmatrix} \mathbf{F}_r \\ \mathbf{F}_e \end{bmatrix} \begin{bmatrix} \mathbf{F}_r^\top & \mathbf{F}_e^\top \end{bmatrix} \tag{2}$$

is any decomposition of the PSD kernel matrix $\mathbf{K}$, and the factor $\mathbf{F}_r \in \mathbf{R}^{n\times k}$ and $\mathbf{F}_e \in \mathbf{R}^{m\times k}$ can be considered as "virtual samples" whose dimensionality k is the rank of $\mathbf{K}$.

Proof: The dual of kernel SVM optimization (1) is

$$\min_{\alpha} \ \frac{1}{2}\alpha^\top \mathbf{Q}_{\text{rr}}\alpha - \sum \alpha_i$$
$$\text{subject to } 0 \le \alpha_i \le C, \quad \sum \alpha_i y_i = 0$$
$$\mathbf{Q}_{\text{rr}} = \mathbf{K}_{\text{rr}} \odot \left(\mathbf{y}_r\mathbf{y}_r^\top\right) \tag{3}$$

where $\odot$ is the entrywise product between matrices and α is the Lagrangian multipliers. Prediction on the test data is

$$\hat{\mathbf{y}}_e = \mathbf{K}_{\text{er}} \cdot (\alpha \odot \mathbf{y}_r). \tag{4}$$

Let $\mathbf{F} = [\mathbf{F}_r^\top \ \mathbf{F}_e^\top]^\top$ and $\mathbf{F}_i$ be the ith column in $\mathbf{F}^\top$. Assume we train a linear SVM using $\mathbf{F}_r$ and $\mathbf{y}_r$, with the primal form

$$\min_{\overline{\mathbf{w}},\overline{\xi}_i,b} \ \frac{1}{2}\|\overline{\mathbf{w}}\|^2 + C\sum \overline{\xi}_i \qquad y_i(\overline{\mathbf{w}}^\top \mathbf{F}_i + b) \ge 1 - \overline{\xi}_i. \tag{5}$$

The dual form of (5) is

$$\min_{\overline{\alpha}} \ \frac{1}{2}\overline{\alpha}^\top \overline{\mathbf{Q}}_{\text{rr}}\overline{\alpha} - \sum_i \overline{\alpha}_i$$
$$\text{subject to } 0 \le \overline{\alpha}_i \le C, \sum \overline{\alpha}_i y_i = 0$$
$$\overline{\mathbf{Q}}_{\text{rr}} = \left(\mathbf{F}_r\mathbf{F}_r^\top\right) \odot \left(\mathbf{y}_r\mathbf{y}_r^\top\right). \tag{6}$$

Then, the prediction on $\mathbf{F}_e$ is

$$\bar{\hat{\mathbf{y}}}_e = \mathbf{F}_e^\top \mathbf{F}_r \cdot (\overline{\alpha} \odot \mathbf{y}_r). \tag{7}$$

Note that $\mathbf{F}_r\mathbf{F}_r^\top = \mathbf{K}_{\text{rr}}$ (2), therefore (3) and (6) are equivalent and lead to the same optimal solution $\alpha^* = \overline{\alpha}^*$. Plugging the optimal solutions into (4) and (7), and note that the fact $\mathbf{F}_e^\top\mathbf{F}_r = \mathbf{K}_{\text{er}}$ (2), we can see that the prediction in (4) and (7) is the same, i.e., $\hat{\mathbf{y}}_e = \bar{\hat{\mathbf{y}}}_e$. For example, the kernel SVM (1) and the linear SVM (5) are equivalent. □

Proposition 1 states that any kernel SVM can be cast as an equivalent linear SVM by decomposition $\mathbf{K} = \mathbf{F}\mathbf{F}^\top$ (2), where $\mathbf{F}$ serves as an empirical kernel map or *virtual samples*.

The decomposition (2) always exists because the kernel matrix is PSD. When only training data is used, the decomposition

$$\mathbf{K}_{\mathrm{rr}} = \mathbf{F}_r \mathbf{F}_r^\top \tag{8}$$

allows us to recover the Lagrangian multipliers in the decision function (3). We will focus on (8) in the rest of this paper.

Motivated by Proposition 1, we propose to train kernel SVM on large-scale data in two steps: 1) transform kernel SVM to a linear one using kernel matrix decomposition and 2) efficiently solve the resultant linear SVM. Obviously, efficiently obtaining the empirical kernel map $\mathbf{F}_r$ (8) is the key to the success. We will discuss how to achieve this and how it would affect the SVM solution in Section III.

III. Low-Rank Linearized SVM

Kernel matrix is the key building block of SVM, its entries represent inner products between samples in feature space. This avoids explicitly computing the map $\psi(\mathbf{x})$ (which can be infinite dimensional). Such "kernel trick" enables SVM to produce nonlinear concepts at the cost of manipulating the $n \times n$ kernel matrix. In comparison, linear SVM uses an explicit map $\psi(\mathbf{x}) = \mathbf{x}$ that offers great potential in efficiency.

Proposition 1 shows that any exact decomposition of the kernel matrix can preserve the dot products among $\psi(\mathbf{x}_i)$'s via *empirical kernel map* $\mathbf{F}_i$'s as $\mathbf{K}_{ij} = \langle \psi(\mathbf{x}_i), \psi(\mathbf{x}_j) \rangle = \langle \mathbf{F}_i, \mathbf{F}_j \rangle$. This is the key to transforming a kernel SVM into an explicit linear counterpart. It bridges the gap between kernel and linear SVM and opens the possibility of training large-scale kernel SVM by advanced linear SVM solvers.

Therefore, the key question is to efficiently obtain an appropriate decomposition (8). Given an $n \times n$ kernel matrix with the eigen-decomposition $\mathbf{K}_{\mathrm{rr}} = \mathbf{U}_r \Lambda_r \mathbf{U}_r^\top$, where Λ_r is a diagonal matrix where diagonal entries are eigenvalues $\Lambda_r(ii) = \lambda_i$ in a descending order and $\mathbf{U}_r \in \mathbf{R}^{n \times n}$ are the corresponding eigenvectors. $\mathbf{F}_r$ in (8) can be chosen as $\mathbf{F}_r = \mathbf{U}_r \Lambda_r^{1/2}$. Theoretically, the eigen-decomposition provides the optimal rank-k approximation of the kernel matrix

$$\min_{\mathrm{rank}(\widetilde{\mathbf{K}}_{\mathrm{rr}})=k} \|\mathbf{K}_{\mathrm{rr}} - \widetilde{\mathbf{K}}_{\mathrm{rr}}\|_F^2 = \sum_{i=k+1}^{n} \lambda_i^2 \tag{9}$$

where $\widetilde{\mathbf{K}}_{\mathrm{rr}}$ is the approximate rank-k matrix. In other words, for a given number k, the feature map

$$\mathbf{F}_r^{(k)} = \mathbf{U}_r^{(k)} (\Lambda_r^{(k)})^{1/2} \tag{10}$$

composed of top eigenvectors/values is the optimal since the inner product it recovers is the closest to $\mathbf{K}_{\mathrm{rr}}$ among all rank-k approximations. The difference equals sum of squared minimum $n - k$ eigenvalues as shown in (9).

However, exact computation of top k eigenvalues/eigenvectors requires $O(n^2)$ space and $O(n^2 k)$ time, which can be costly. We are interested in the Nytström method that has gained great popularity in scaling up kernel-based approaches [22], [31]. Given training data $\mathbf{X}_r$ and the kernel matrix $\mathbf{K}_{\mathrm{rr}}$, the Nyström method selects a landmark set of k samples $\mathcal{Z}$ and approximates the kernel matrix as

$$\widetilde{\mathbf{K}}_{\mathrm{rr}} = \mathbf{K}_{\mathrm{rz}} \mathbf{K}_{\mathrm{zz}}^{-1} \mathbf{K}_{\mathrm{rz}}^\top \tag{11}$$

where $\mathbf{K}_{\mathrm{rz}} \in \mathbf{R}^{n \times k}$ is the kernel matrix on $\mathbf{X}_r$ and $\mathbf{X}_z$, and $\mathbf{K}_{\mathrm{zz}} \in \mathbf{R}^{k \times k}$ is the kernel matrix on $\mathcal{Z}$.

Let the eigen-decomposition of $\mathbf{K}_{\mathrm{zz}}$ be $\mathbf{U}_z \Lambda_z \mathbf{U}_z^\top$, then the Nyström method approximates the optimal kernel map (10) as

$$\widetilde{\mathbf{K}}_{\mathrm{rr}} = \widetilde{\mathbf{F}}_r \widetilde{\mathbf{F}}_r^\top, \text{ where } \widetilde{\mathbf{F}}_r = \mathbf{K}_{\mathrm{rz}} \mathbf{U}_z \Lambda_z^{-1/2}. \tag{12}$$

The rank-k approximation by Nyström method (11) provides a natural approximation to the optimal rank-k kernel map $\mathbf{F}_r^{(k)}$ (10). To see this, consider enlarging the landmark set $\mathcal{Z}$ to $\mathbf{X}_r$: then $\mathbf{U}_z \to \mathbf{U}_r$, $\Lambda_z \to \Lambda_r$, $\mathbf{K}_{\mathrm{rz}} \to \mathbf{K}_{\mathrm{rr}}$. As a result, when $|\mathcal{Z}| \to n$, $\mathbf{K}_{\mathrm{rz}} \mathbf{U}_z \Lambda_z^{-1/2} = \mathbf{K}_{\mathrm{rr}} \mathbf{U}_r \Lambda_r^{-1/2} = \mathbf{U}_r \Lambda_r \mathbf{U}_r^\top \mathbf{U}_r \Lambda_r^{-1/2} = \mathbf{U}_r \Lambda_r^{1/2}$. Namely, $\widetilde{\mathbf{F}}_r \to \mathbf{F}_r$.

Extensive work has been devoted to bound the approximation error of the Nytström method in Frobenius norm [27], [31]

$$\mathcal{E}_f = \|\mathbf{K}_{\mathrm{rr}} - \widetilde{\mathbf{K}}_{\mathrm{rr}}\|_F \tag{13}$$

and the spectral norm [25]

$$\mathcal{E}_2 = \|\mathbf{K}_{\mathrm{rr}} - \widetilde{\mathbf{K}}_{\mathrm{rr}}\|_2. \tag{14}$$

Usually the error bounds consist of two terms: the rank-k approximation error via exact eigen-decomposition and the slackness of the approximation to the eigen-decomposition.

Next, we analyze how Nyström low-rank approximation affects the SVM solution. In particular, we focus on the gap between the SVM solutions trained based on the optimal and the Nyström-based kernel maps.

Theorem 1: Let $\mathbf{w}$ and $\mathbf{w}'$ be solutions of (1) using optimal and Nyström-based rank-k empirical kernel maps, $\mathbf{F}_r^{(k)}$ and $\widetilde{\mathbf{F}}_r^{(k)}$ in (10) and (12). Then, $\|\mathbf{w} - \mathbf{w}'\|^2$ is bounded as

$$2C^2 \rho^{\frac{1}{2}} \left[k e_f^{\frac{1}{4}} + \left(\lambda_1 \mathrm{tr}(A) + k e_2 \mathrm{tr}\left(\widetilde{\Lambda}_{(k)}^{-1}\right) \right) \left(e_f^{\frac{1}{4}} + \mathrm{tr}\left(\Lambda_{(k)}^2\right)^{1/4} \right) \right]$$

where $A \in \mathbf{R}^{k \times k}$ is a diagonal matrix with entries $A_{ii} = \max((1/\Lambda_{ii}) + (1/\widetilde{\Lambda_{ii}}), (3/\widetilde{\Lambda_{ii}}) - (1/\Lambda_{ii}))$, $\Lambda = \mathrm{diag}(\lambda_1, \ldots, \lambda_n)$, and $\widetilde{\Lambda} = \mathrm{diag}(\tilde{\lambda}_1, \ldots, \tilde{\lambda}_k)$ are exact and approximate (by Nyström) eigenvalues (sorted in descending order) of the kernel matrix

$$e_f = \left(\sum_{i=k+1}^{n} \lambda_i^2 \right)^{1/2} + \mathcal{E}_f \tag{15}$$

$$e_2 = \lambda_{k+1} + \mathcal{E}_2 \tag{16}$$

and $\mathcal{E}_f$ and $\mathcal{E}_2$ are known error bounds [(13) and (14)] on the gap between the Nyström low-rank approximation and the original kernel matrix; ρ is a kernel dependent constant.

The proof can be found in [30]. Theorem 1 shows that the gap between SVM solutions based on the approximate and optimal rank-k kernel maps is bounded by Nyström low-rank approximation error. The lower the approximation error is the better the classification accuracy. Unlike the error bound in [25] that applies to general approximations of the kernel matrix, our analysis focuses on the impact of a clustering-based Nystöm low-rank approximation on the SVM solution. Therefore, we are interested in obtaining accurate low-rank approximation through the Nyström method, which in turn depends crucially on the sampling scheme adopted. In this paper, we will resort to the k-means-based sampling [23] that has shown to significantly outperform random sampling [22], greedy schemes, and probabilistic sampling [27]. We adopt a fast approximate k-means using only a small number of iterations in our experiments, therefore the computational complexity is linear with sample size and dimension. After linearizing the kernel SVM, we use SBM [29] to train the linear SVM on large-scale data that cannot fit in memory, where the data is chunked into nonoverlapping subsets and sequentially loaded into memory.

We summarize our proposed approach in Algorithm 1 and termed as low-rank linearized SVM (LLSVM). LLSVM has a constant space and linear time complexity in terms of sample size. Predicting of LLSVM is also very efficient.

Algorithm 1 LLSVM

```
Training
Input: training data set ${\mathbf{X}_r, \mathbf{y}_r}$, kernel $\kappa$, regularization parameter $C$, landmark set size $k$;
Output: model $\hat{\mathbf{w}} \in \mathbf{R}^{k\times 1}$, mapping $\mathbf{M} \in \mathbf{R}^{k\times k}$;
1: select $k$ landmark points $\mathcal{Z}$ from $\mathbf{X}_r$;
2: compute $\mathbf{K}_{zz}$, eigen-decomposition $\mathbf{K}_{zz} = \mathbf{U}_z\Lambda_z\mathbf{U}_z^\top$;
3: compute $\mathbf{M} = \mathbf{U}_z\Lambda_z^{-1/2}$;
4: compute $\mathbf{K}_{\mathrm{rz}}$, train linear SVM on $\mathbf{K}_{\mathrm{rz}}\mathbf{M}$ by SBM and obtain model $\hat{\mathbf{w}}$;
Prediction
Input: test data $\mathbf{X}_e$, model $\hat{\mathbf{w}}$, mapping matrix $\mathbf{M}$;
Output: predicted labels $\hat{\mathbf{y}}_e$;
1: compute $\mathbf{K}_{\mathrm{ez}}$, and predict by $\hat{\mathbf{y}}_e = \hat{\mathbf{w}}^\top\ \mathbf{K}_{\mathrm{ez}}\mathbf{M}$;
```

Choice of the landmark size k: The choice of k is a tradeoff between scalability and accuracy. Typically, the larger the k, the lower the approximation error, and so the more accurate the resultant predictor is expected to be (see analysis in Section III). In the meantime, the computational cost will also be more demanding. In our empirical evaluations, for data set whose size is around hundreds of thousands, we usually choose k as a few thousands. Bach [32] discussed in more detail the choice of the rank parameter k to avoid the loss of prediction accuracy compared to the full-kernel matrix.

IV. Incorporating Semisupervised Metric Learning for Low-Rank Linearization

One limitation of the Nyström method is that it is purely unsupervised. Therefore, the resultant kernel map may not always be well aligned to the learning task (e.g., when kernel parameters are selected inappropriately). In this section, we incorporate the class label in the Nyström method such that the geometry of the resultant low-rank kernel embedding becomes more consistent with the class labels.

Remind that the training data $\mathbf{X}_r$ will be transformed to $\widetilde{\mathbf{F}}_r = \mathbf{K}_{\mathrm{rz}}\mathbf{U}_z\Lambda_z^{-1/2}$ in our kernel map. Then the similarity between two data points $\mathbf{x}_i$ and $\mathbf{x}_j$ becomes $\kappa(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{K}_{\mathrm{rz}}\mathbf{U}_z)_i\Lambda_z^{-1}(\mathbf{K}_{\mathrm{rz}}\mathbf{U}_z)_j^T$, where $(\mathbf{K}_{\mathrm{rz}}\mathbf{U}_z)_i \in \mathbf{R}^{1\times k}$ denotes the ith row of the matrix $\mathbf{K}_{\mathrm{rz}}\mathbf{U}_z$. The diagonal of Λ_z^{-1} is reweighting coefficients on different dimensions of $\mathbf{K}_{\mathrm{rz}}\mathbf{U}_z$. Here, instead of using the old eigenvalues Λ_z, we aim to learn a new eigenvalue matrix $\mathbf{\Sigma}_z$ using the class labels. Inspired by the idea of metric learning [33], the goal is to improve the interclass separability along with the intraclass compactness in the new feature map $\widetilde{\mathbf{F}}'_r = \mathbf{K}_{\mathrm{rz}}\mathbf{U}_z\mathbf{\Sigma}_z^{-1/2}$.

Let $\mathbf{z}_i$ be the feature map for the ith training example (or the ith row in $\widetilde{\mathbf{F}}'_r$). Then, the interclass separability can be measured as the sum of squared pairwise distances between $\mathbf{z}_i$'s from different classes, as $\sum_{y_i\neq y_j}\|\mathbf{z}_i - \mathbf{z}_j\|^2$; similarly, the intraclass tightness can be measured as by the sum of squared pairwise distances between $\mathbf{z}_i$'s from the same class, as $\sum_{y_i=y_j}\|\mathbf{z}_i - \mathbf{z}_j\|^2$. By combining these two terms, and suppose, we only consider using a subset[1] of labeled samples $\{\mathbf{X}_{\mathscr{L}}, \mathbf{y}_{\mathscr{L}}\} \in \{\mathbf{X}_r, \mathbf{y}_r\}$ to improve the kernel map, we need to minimize

$$\begin{aligned}\sum_{i,j\in\mathscr{L}}\|\mathbf{z}_i - \mathbf{z}_j\|^2\mathbf{A}_{ij} &= \mathbf{Tr}\Big(\Big(\mathbf{K}_{\mathrm{rz}}\mathbf{U}_z\mathbf{\Sigma}_z^{-\frac{1}{2}}\Big)^\top\mathbf{L}\Big(\mathbf{K}_{\mathrm{rz}}\mathbf{U}_z\mathbf{\Sigma}_z^{-\frac{1}{2}}\Big)\Big)\\ &= \mathbf{Tr}\big(\mathbf{U}_z{}^\top\mathbf{K}_{\mathrm{rz}}{}^\top\mathbf{L}\mathbf{K}_{\mathrm{rz}}\mathbf{U}_z\mathbf{\Sigma}_z^{-1}\big) \qquad (17)\end{aligned}$$

where $\mathbf{A}_{ij} = 1$ if $y_i = y_j$, and $\mathbf{A}_{ij} = -a$ if $y_i \neq y_j$, parameter a is used to balance the tradeoff between with-in class and between-class constraints, $\mathbf{L}$ is the Laplacian matrix of $\mathbf{A}$, and $\mathbf{Tr}(\mathbf{M})$ denotes the trace of matrix $\mathbf{M}$.

We regularize the learned $\mathbf{\Sigma}_z$ such that it is somewhat close to the original Nyström solution Λ_z, which guarantees that the low-rank structure of the kernel matrix is well preserved. Therefore, the optimization is formulated as

$$\begin{aligned}&\min_{\mathbf{\Sigma}}\ \mathbf{Tr}\big(\mathbf{U}_z{}^T\mathbf{K}_{\mathrm{rz}}{}^T\mathbf{L}\mathbf{K}_{\mathrm{rz}}\mathbf{U}_z\mathbf{\Sigma}_z^{-1}\big) + \lambda\big\|\mathbf{\Sigma}_z^{-1} - \Lambda_z{}^{-1}\big\|^2.\\ &\text{subject to}\ \ (\mathbf{\Sigma}_z)_{ii} \geq 0 i = 1, 2, \ldots, k. \qquad (18)\end{aligned}$$

Let vector $\mathbf{c}$ denote the diagonal of the matrix $\mathbf{U}_z{}^\top\mathbf{K}_{\mathrm{rz}}{}^\top\mathbf{L}\mathbf{K}_{\mathrm{rz}}\mathbf{U}_z$, i.e., $c_i = (\mathbf{U}_z{}^\top\mathbf{K}_{\mathrm{rz}}{}^\top\mathbf{L}\mathbf{K}_{\mathrm{rz}}\mathbf{U}_z)_{ii}$, vector $\boldsymbol{\mu}$ denote the diagonal of $\mathbf{\Sigma}_z^{-1}$, and vector $\boldsymbol{\mu}_\mathbf{0}$ denote the diagonal of Λ_z^{-1}. Then, the objective in (18) can be rewritten as $\min_{\boldsymbol{\mu}}\mathbf{c}^\top\boldsymbol{\mu} + \lambda\|\boldsymbol{\mu} - \boldsymbol{\mu}_\mathbf{0}\|^2$ that is equivalent to

$$\begin{aligned}&\min_{\boldsymbol{\mu}}\ \ \Big\|\boldsymbol{\mu} - \Big(\boldsymbol{\mu}_\mathbf{0} - \frac{\mathbf{c}}{2\lambda}\Big)\Big\|^2 + \text{const}\\ &\text{subject to}\ \mu_i \geq 0, i = 1, 2, \ldots, k. \qquad (19)\end{aligned}$$

Since (19) is a constrained optimization, by adding Lagrangian multipliers $\beta_i, i = 1, \ldots, k$, with $\beta_i \geq 0$, we have

$$L(\boldsymbol{\mu}, \boldsymbol{\beta}) = \Big\|\boldsymbol{\mu} - \Big(\boldsymbol{\mu}_\mathbf{0} - \frac{\mathbf{c}}{2\lambda}\Big)\Big\|^2 - \sum_{i=1}^{k}\beta_i\mu_i. \qquad (20)$$

According to Karush–Kuhn–Tucker condition, any local minimizer $\boldsymbol{\mu}^*$ must satisfy the followings:

$$\begin{cases}2\left(\mu_i^* - \left((\mu_0)_i - \dfrac{c_i}{2\lambda}\right)\right) - \beta_i = 0,\\ \beta_i\mu_i^* = 0.\end{cases} \qquad (21)$$

Since μ_i^* and β_i are both nonnegative, $\beta_i\mu_i^* = 0$ is equivalent if $\mu_i^* > 0$, then $\beta_i = 0$. Hence, (21) can be rewritten as

[1]For computational efficiency, we only utilize labels of a fraction of training samples to improve the low-rank kernel map $\widetilde{\mathbf{F}}'_r$. In the SVM training phase, we will use labels of all the training samples to obtain a classifier.

$$\begin{cases} \mu_i^* = \Big((\mu_0)_i - \dfrac{c_i}{2\lambda}\Big), & \text{if } \mu_i^* > 0 \\ \beta_i = -2\Big((\mu_0)_i - \dfrac{c_i}{2\lambda}\Big) \geq 0, & \text{if } \mu_i^* = 0. \end{cases} \tag{22}$$

Therefore, we obtain the global optimal solution for (19)

$$\mu_i^* = \max\Big(0, (\mu_0)_i - \frac{c_i}{2\lambda}\Big). \tag{23}$$

After obtaining the new eigenvalue Σ_z, we compute the new feature map as $\widetilde{\mathbf{F}}'_r = \mathbf{K}_{\text{rz}}\mathbf{U}_z\boldsymbol{\Sigma}_z^{-1/2}$. Then, a linear SVM is trained on $\widetilde{\mathbf{F}}'_r$ together with the labels $\mathbf{y}_r$. We name this algorithm LLSVM^2 and summarized it in Algorithm 2. Since the number of labels used $|\mathscr{L}|$ is typically chosen as a small constant in comparison with the sample size, and c_i's in (23) are computed as $c_i = (\mathbf{U}_z{}^T\mathbf{K}_{\text{rz}}{}^T\mathbf{L}\mathbf{K}_{\text{rz}}\mathbf{U}_z)_{ii}$ and the Laplacian matrix $\mathbf{L}$ is equal to $\mathbf{D} - \mathbf{y}\mathbf{y}^T$ in our case, the time and space complexities of the whole algorithm are still in linear with respect to the sample size.

Algorithm 2 LLSVM^2 (Training Stage)

```
1: select $k$ landmark points $\mathcal{Z}$ from $\mathbf{X}_r$;
2: compute $\mathbf{K}_{zz}$, eigen-decomposition $\mathbf{K}_{zz} = \mathbf{U}_z\Lambda_z\mathbf{U}_z^\top$;
3: compute new eigenvalues $\boldsymbol{\Sigma}_z$ (23) using $\{\mathbf{X}_{\mathscr{L}}, \mathbf{y}_{\mathscr{L}}\}$
4: compute $\mathbf{M} = \mathbf{U}_z\boldsymbol{\Sigma}_z^{-1/2}$;
5: compute $\mathbf{K}_{\text{rz}}$, train linear SVM on $\mathbf{K}_{\text{rz}}\mathbf{M}$ by SBM.
```

V. Related Methods

A. Nyström Methods

The Nyström method is widely used for scaling up kernel methods, such as large kernel matrix inversion [22], the kernel eigenvectors [34], and manifold learning [24]. Recently, Cortes *et al.* [25] study the impact of kernel approximation on the accuracy of kernel-based learning algorithms, and also in the specific context of Nyström low-rank approximation. Their theoretical analysis is based on the following kernel map $\Phi(\mathbf{x}) = (\mathbf{K}^\dagger)^{1/2}[k(\mathbf{x}_1, \mathbf{x}), k(\mathbf{x}_2, \mathbf{x}), \ldots, k(\mathbf{x}_n, \mathbf{x})]^\top$, where $\mathbf{K}^\dagger$ denotes the pseudoinverse. As shown in (21), this is equivalent to using $(\mathbf{K}^\dagger)^{1/2}\mathbf{K} = \mathbf{K}^{1/2}$ as "virtual samples" in a linear SVM. Since $\mathbf{K}^{1/2} = \mathbf{U}\Lambda^{1/2}\mathbf{U}^\top$ is an $n \times n$ matrix, it requires $O(n^2)$ space. Similar complexity arises even if approximate eigenvectors are used. In comparison, our proposed empirical kernel map (12) is an $n \times k$ matrix and $k \ll n$. So greatly improves the efficiency of SVM training.

B. Reduced SVM

As shown in [28] and [35], kernel SVM can be (approximately) transformed to linear SVM in the context of RSVM. In particular, RSVM restricts the model to be spanned by only a small subset of samples $\mathcal{Z}$, $\mathbf{w} = \sum_{i\in\mathcal{Z}} y_i\alpha_i\psi(\mathbf{x}_i)$. After adding a penalty term b^2, the optimization problem of RSVM is written as

$$\begin{aligned} \min_{\mu_z,\xi}\ & \frac{1}{2}\big(\mu_z^\top\mathbf{K}_{\text{zz}}\mu_z\big) + \frac{1}{2}b^2 + C\sum_{i=1}^n \xi_i \\ \text{subject to } & \mathbf{K}_{\text{rz}}\mu_z + b\mathbf{y}_r \geq e - \xi \end{aligned} \tag{24}$$

where $\mu_z = \mathbf{y}_z \odot \alpha_z$, $\mathbf{y}_z$ is the training label, α_z is the Lagrangian multipliers of $\mathcal{Z}$, $\mathbf{K}_{\text{zz}}$ is the kernel matrix on $\mathcal{Z}$, $\mathbf{K}_{\text{rz}}$ is the kernel matrix between $\mathbf{X}_r$ and $\mathcal{Z}$, and e is a vector of all 1's.

TABLE I
Empirical Kernel Map in Different Methods ($|\mathcal{Z}| = k \ll n$)

Method	kernel map	feature dimensionality
(Cortes *et al.*, 2010)	$\mathbf{K}_{rr}^{1/2}$	$n \times n$
RSVM	$\mathbf{K}_{rz}$	$n \times k$
LLSVM (ours)	$\mathbf{K}_{rz}\mathbf{U}_z\boldsymbol{\Lambda}_z^{-1/2}$	$n \times k$

Solving the quasi-peak (QP) (24) can be difficult due to the Hessian matrix $\mathbf{K}_{\text{zz}}$. Lee and Mangasarian [35] proposed to drop $\mathbf{K}_{\text{zz}}$ from (24) which reduces (24) to a linear SVM and can be efficiently solved. However, dropping $\mathbf{K}_{\text{zz}}$ leads to inferior accuracy than exactly solving (24). We will use low-rank linearization to efficiently obtain an exact solution of (24).

Proposition 2: Consider the RSVM (24). Perform eigen-decomposition $\mathbf{K}_{\text{zz}} = \mathbf{U}_z\Lambda_z\mathbf{U}^\top$. Then, (24) can be solved exactly by a linear SVM using $\mathbf{K}_{\text{rz}}\mathbf{U}_z\Lambda_z^{-1/2}$ as training samples. Testing on $\mathbf{X}_e$ can be performed by applying the learned linear model on $\mathbf{K}_{\text{ez}}\mathbf{U}_z\Lambda_z^{-1/2}$.

Proof: Given $\mathbf{K}_{\text{zz}} = \mathbf{U}_z\Lambda_z\mathbf{U}^\top$, define a linear transform $\beta_z = \Lambda_z^{1/2}\mathbf{U}_z^\top\mu_z$. Note that Λ_z is a diagonal matrix whose diagonal entries are the eigenvalues. Λ_z is invertible if the eigenvalues are all strictly positive. Note that $\mathbf{U}_z\mathbf{U}_z^\top = \mathbf{U}_z^\top\mathbf{U}_z = \mathbf{I}$. So, we can invert the above transform as $\mu_z = \mathbf{U}_z\Lambda_z^{-1/2}\beta_z$. Plugging μ_z into (24), we have the problem

$$\begin{aligned} \min_{\beta_z,\xi}\ & \frac{1}{2}\|\beta_z\|^2 + \frac{1}{2}b^2 + C\sum_{i=1}^n \xi_i \\ \text{subject to } & \big(\mathbf{K}_{\text{rz}}\mathbf{U}_z\Lambda_z^{-1/2}\big)\beta_z + b\mathbf{y}_r \geq e - \xi \end{aligned} \tag{25}$$

which is equivalent to training a linear SVM on $\mathbf{K}_{\text{rz}}\mathbf{U}_z\Lambda_z^{-1/2}$. The prediction of RSVM (24) on testing set is $\mathbf{K}_{\text{ez}}\mu_z$. Since $\mu_z = \mathbf{U}_z\Lambda_z^{-1/2}\beta_z$, the prediction is rewritten as $(\mathbf{K}_{\text{ez}}\mathbf{U}_z\Lambda_z^{-1/2})\beta_z$. Namely, it applies the learned model β_z on "virtual" test samples $\mathbf{K}_{\text{ez}}\mathbf{U}_z\Lambda_z^{-1/2}$. □

Proposition 2 provides a new way to efficiently solve RSVM. When the subkernel matrix $\mathbf{K}_{\text{zz}}$ is strictly positive definite, the solution is exact, in comparison with those used in [28] and [35]. Our method applies to any SVM formulation using PSD kernel, while RSVM is designed for a specific type of SVM.

Proposition 2 has the same form as the LLSVM proposed in Section III. Therefore, tremendous work on landmark selection in the Nyström method [22], [23] can be used for efficient RSVM. In comparison, the original RSVM simply uses random sampling, or couples the subset selection with sparse SVM training which can be quite expensive on large data [36]. Another interesting observation is that the resultant problem of (24) after removing the Hessian $\mathbf{K}_{\text{zz}}$ is equivalent to a linear SVM trained on $\mathbf{K}_{\text{rz}}$. So, training samples in RSVM is re-expressed by their kernel similarity with k landmark points. We list the empirical kernel maps (and their dimensions) used in three related methods [25], [28], [35] in Table I.

C. Supervised Low-Rank Approximation

Cholesky with side information method [37] exploits class labels in transductive kernel low-rank approximation. It iteratively selects columns of the kernel matrix to reduce matrix approximation and linear prediction errors. Recently, Zhang *et al.* [38] proposed a semisupervised inductive kernel low-rank decomposition method, which use class labels to facilitate learning of the dictionary kernel similarity matrix $\mathbf{K}_{zz}$ on a preselected subset of landmark points as shown in (11). Our approach (LLSVM^2) focuses on learning the spectrum of $\mathbf{K}_{zz}$, which has a closed form solution and is more efficient.

D. Kernel Learning

Kernel learning aims at computing a kernel matrix that is better aligned with the class labels to improve the generalization performance. A popular approach is to learn a convex combination of multiple base kernels, such as the semidefinite programming framework [39]. Kulis *et al.* [40] proposed to learn a low-rank kernel matrix using Bregman divergence. Recently, Mao *et al.* [41] used data-dependent prior based on an ensemble of kernel predictors to improve multiple kernel learning. These methods are mostly transductive while we focus more on inductive setting to handle new coming data.

VI. Experiments

In Section VI-A, we compare LLSVM with eight state-of-the-art algorithms for training large-scale SVM on five big data sets. In Section VI-B, we show that LLSVM^2 can further improve the performance of LLSVM, in particular, in terms of the stability of algorithm performance with regard to the choice of hyperparameters.

A. Performance of LLSVM

We compare LLSVM with the following algorithms:

1) *SBM:* A linear SVM solver using limited memory [29];
2) *Libsvm:* A popular exact solver for kernel SVM [5];
3) *Poly2:* A fast solver for degree-2 polynomial SVM [17];
4) *Core vector machine (CVM):* an approximate SVM solver [6];
5) *Lasvm:* An online approximate SVM solver [7];
6) *Adaptive multihyperplane machine (AMM):* it approximates nonlinear classifier by multiple linear classifiers [42], [43];
7) *Random kitchen sinks (RKS):* A fast SVM solver via randomized Fourier features [19];
8) *Fastfood:* A solver that accelerates RKS [44].

1) Data Sets: We use five large benchmark data sets[2] as given in Table II to evaluate the performance. The original multiclass mnist8m is transferred into binary mnist8m-b by using round digits (3, 6, 8, 9, 0) versus nonround digits (1, 2, 4, 5, 7). Ncheckerboard is a noisy version of 4×4 Checkerboard with 20% randomly shuffled labels (the test part is noisy-free).

[2]Data from http://www.csie.ntu.edu.tw/c̃jlin/libsvmtools/data sets/

TABLE II
SUMMARY OF DATA SETS

data set	training size	testing size	# of features	# of classes	file size
IJCNN	49,990	91,701	22	2	20.9M
webspam	280,000	70,000	254	2	327M
covType	464,810	116,202	54	2	68.1M
Ncheckerboard	800,000	20,000	2	2	22.7M
mnist8m-b	8,000,000	100,000	784	2	18G

2) Setup: We specify radial basis function (RBF) kernel here for Libsvm, CVM, Lasvm, RKS, and Fastfood. In LLSVM, we set $p\% = 75\%$ and the subset size equal to 20 k for the SBM algorithm, and we only run k-means clustering using the first 20 k samples for efficiency. We also use the same SBM implementation to solve the resultant linear SVM problem in RKS and Fastfood. We set the number of scans on the data for RKS, Fastfood, and LLSVM to 1 (i.e., a single pass). It is worth to note that as the number of scans increases the accuracy improve while the training time also increases.

3) Parameter Setting: The features for all the data sets are linearly scaled to [0, 1]. We set the kernel parameter γ in RBF kernel $\kappa(\mathbf{x}_i, \mathbf{x}_j) = \exp(-\|\mathbf{x}_i - \mathbf{x}_j\|^2/\gamma)$ to $\{2^{-5}, 2^{-4}, \ldots, 2^5\}$, the regularizer parameter C to $\{2^{-5}, 2^{-4}, \ldots, 2^5\}$. The number of landmark points k for LLSVM (i.e., the number basis functions in RKS and Fastfood) is given in the last row in Table III. We set γ in polynomial kernel $\kappa(\mathbf{x}_i, \mathbf{x}_j) = (\gamma \mathbf{x}_i^T \mathbf{x}_j + 1)^2$ in Poly2 to $\{2^{-5}, 2^{-4}, \ldots, 2^5\}$. We also evaluate our proposed method using polynomial kernel. We set the parameter γ in $\kappa(\mathbf{x}_i, \mathbf{x}_j) = (\gamma \mathbf{x}_i \mathbf{x}_j^T + 1)^d$ to $\{2^{-4}, 2^{-3} \ldots, 2^4\}$ and set the degree parameter d to $\{2, 4, 6, 8\}$. For other parameters, we use default settings in their implementations. The optimal hyperparameters for all algorithms are selected by tenfold cross validation. Results for nondeterministic algorithms are based on five repetitions of experiments.

4) Results: We summarize the error rate and training time (excluding I/O time) of all the algorithms in Table III. Our experiments are performed on a PC with Intel(R) Core i7-2600 3.40-GHz CPU and 4GB RAM. '–' is used to denote that the algorithm cannot be completed within 24 h. With regards to accuracy, LLSVM achieves significantly lower error rates than linear SVM on all five data sets. With regards to training time, LLSVM is slower than the linear SVM but is still affordable considering our machine configuration. Compared with kernel SVM solvers (Libsvm, CVM, Lasvm), the error rates of LLSVM are comparable to the lowest achievable error rate on each data set (except covType) but LLSVM is much faster. Considering that nearly half of the training samples (around 230 k) becomes support vectors for a kernel SVM in covType task, our error rate using only 4000 landmark points is quite acceptable. More importantly, neither the exact solver (Libsvm) nor the two popular large-scale SVM solvers (CVM and Lasvm) can complete the training on the largest data set mnist8m-b or the noisiest data set NcheckBoard within 24 h due to the curse of kernelization; while LLSVM can easily scale up on such data sets (actually even on arbitrarily large

TABLE III
TRAINING TIME (IN SECONDS/HOURS) AND TEST ERROR (EXCLUDE I/O TIME; '–' MARKS ALGORITHMS THAT CANNOT FINISH WITHIN 24 h)

Algorithms \ Datasets		IJCNN	webspam	covtype	Ncheckerboard	mnist8m-b
Linear	err(%)	7.87±0.01	6.96±0.01	23.75±0.00	48.90±0.54	24.18±0.39
	time	2s	39s	16s	60s	1h
Poly2	err(%)	2.16	1.56	19.91	44.02	–
	time	25s	1.0h	1.0h	14s	–
Libsvm	err(%)	1.31	0.80	–	–	–
	time	30s	2.4h	–	–	–
CVM	err(%)	**1.20**	0.78	**2.50**	–	–
	time	72s	4h	8.3h	–	–
Lasvm	err(%)	1.47	**0.75**	–	–	–
	time	20s	5.6h	–	–	–
AMM	err(%)	2.4±0.11	4.50±0.24	24.02±0.31	35.98±4.09	3.40±0.33
	time	1s	11s	9s	10s	0.3h
RKS (RBF)	err(%)	2.68±0.16	2.69±0.02	16.27±0.07	2.43±0.08	5.40±0.08
	time	51s	620s	538s	110s	2.1h
Fastfood (RBF)	err(%)	3.98±0.20	3.16±0.07	20.49±0.18	2.08±0.03	9.37±0.12
	time	66s	401s	508s	127s	2.2h
LLSVM(polynomial)	err(%)	1.47±0.01	2.34±0.03	23.87±0.03	43.72±0.04	6.80±0.02
	time	135s	859s	1289s	253s	4.7h
LLSVM(RBF)	err(%)	1.24±0.01	1.94±0.04	13.93±0.14	**0.59±0.00**	**2.59±0.01**
	time	123s	860s	1296s	244s	4.5h
	# of landmark points	($k = 3000$)	($k = 4000$)	($k = 4000$)	($k = 1000$)	($k = 3000$)

data sets) using a regular PC. In comparison with AMM, LLSVM gets significantly lower error rates with affordable time. In comparison with RKS and Fastfood, LLSVM also gets lower error rates and is only a few times slower. Fastfood is faster than RKS on computing the kernel mapping, but the overall training time is very similar to RKS because it is dominated by training the linear SVM. In Table III, we can also observe that LLSVM with RBF kernel can be more accurate than with Polynomial kernel. In particular, on the noisiest data set Ncheckerboard, no algorithms with polynomial kernel can achieve a good accuracy, which indicates that polynomial kernel is not suitable for noisy 4×4 XOR problem.

5) Impact of Parameter k: RKS, Fastfood, and LLSVM approximate kernel similarity among samples by creating explicit feature map. The approximation quality highly depends on parameter k (i.e., number of landmark points for LLSVM; number of basis functions for RKS and Fastfood). In Table III, we compare these methods using different choices in k ($k = \{500, 1000, 2000, 3000, 4000, 5000, 6000, 7000, 8000\}$). Fig. 1(a) and (b) shows the error rate with different k. We observe that LLSVM always achieves lower error rate than RKS and Fastfood for the same k, indicating that our method can extract more information with the same number of landmarks given. Fig. 1(c) and (d) shows training time with different k. As expected, the training time for LLSVM grows quadratically with respect to k, and the training time for RKS and Fastfood grows linearly with respect to k. Therefore, using the same number of landmark points, LLSVM can be several time slower than RKS and Fastfood. However, under the same time consumption, LLSVM can be more accurate. In Fig. 1(e) and (f), we show the error rate of different algorithms versus training time. As can be observed, LLSVM typically achieves a lower error rate when consuming the same amount of time compared with RKS and Fastfood methods.

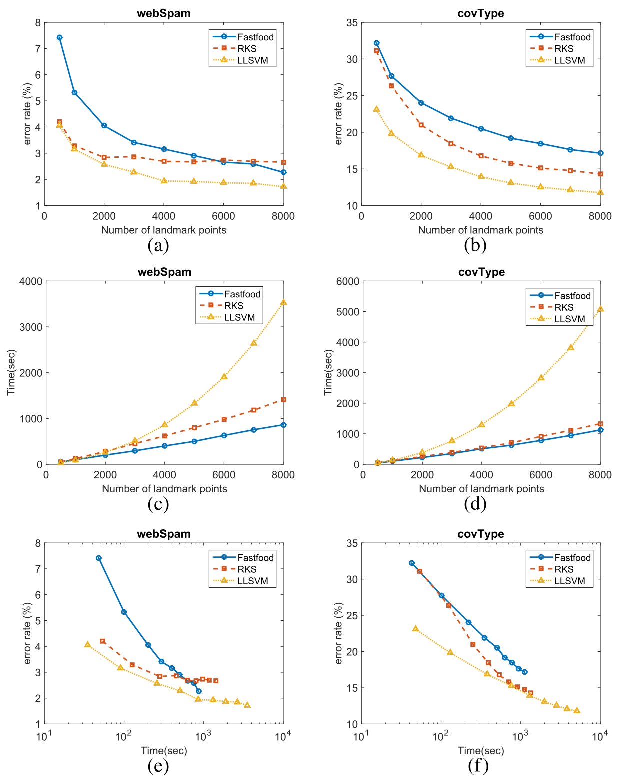


Fig. 1. Classification error rate versus the number of landmark points, training time versus the number of landmark points, and classification error rate versus training time for Fastfood, RKS, and LLSVM, respectively. (a) WebSpam, error rate. (b) CovType, error rate. (c) WebSpam, time. (d) CovType, time. (e) WebSpam. (f) covType.

We summarize the prediction time complexity of different methods in Table IV. The SVM using RBF kernel involves extensive kernel evaluations that in turn grow linearly with the

TABLE IV

PREDICTION COMPLEXITY ON AN INSTANCE

Method	Complexity[a]	Method	Complexity
LinearSVM	$O(c_1 d)$	RKS	$O(c_1 kd)$
RBFSVM	$O(c_2 \|S\|)$	Fastfood	$O(c_1 k log(d))$
Poly2SVM	$O(c_1 d^2)$	LLSVM	$O(c_1 k^2 + c_2 k)$
AMM	$O(c_1 md)$	LLSVM^2	$O(c_1 k^2 + c_2 k)$

[a]$d, |S|, k$ is the number of features,support vectors, and landmark points respectively; n is training sample size and $k \ll n$; m is the number of linear weights in AMM; c_1 is the computing time for a scalar product; c_2 is the computing time for evaluating a RBF kernel function and $c_1 \ll c_2$.

TABLE V

DATA SETS USED FOR EVALUATION OF LLSVM AND LLSVM^2

data set	$n_r/n_e/d/\#cls$	data set	$n_r/n_e/d/\#cls$
DIGIT	1000/500/241/2	GISETTE	6000/1000/5000/2
COIL2	1000/500/241/2	USPS	7291/2007/256/2
SVMGUIDE	3089/4000/4/2	PENDIGIT	7494/3498/16/2

number of support vectors. On the other hand, Linear, Poly2, AMM, RKS, and Fastfood have much lighter prediction burden, which only need a number of scalar product operations. The prediction time complexity of LLSVM is somewhere in between and can be freely controlled by user-defined parameter k. When k is small the kernel evaluation dominates the cost; when k is large the transformation (multiplication with the $k \times k$ mapping matrix $\mathbf{M}$ in Algorithm 1) dominates.

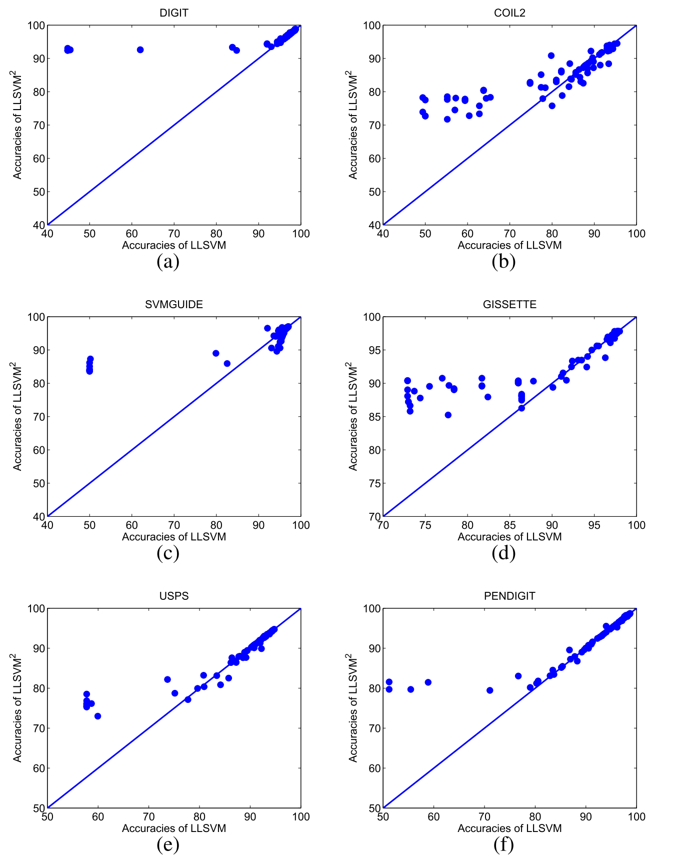


Fig. 2. Classification Accuracies of LLSVM (x-axis) and LLSVM^2 (y-axis) for altogether 77 different combinations of parameters. (a) DIGIT. (b) COIL2. (c) SVMGUIDE. (d) GISETTE. (e) USPS. (f) PENDIGIT.

B. Performance of $LLSVM^2$

1) Stability Comparison Between $LLSVM^2$ and $LLSVM$: In this section, we compare the stability of LLSVM and LLSVM^2 in terms of kernel and regularization parameters. We have adopted some median sized benchmark data sets as summarized in Table V. The original multiclass data sets USPS and PENDIGIT were converted to binary using round digits versus nonround digits. For LLSVM^2, we randomly choose 100 labeled samples from the training set as the $\{\mathbf{X}_{\mathscr{L}}, \mathbf{y}_{\mathscr{L}}\}$ to improve the kernel map. We use the rest training data for validation to select optimal λ in (19). The prediction accuracy is evaluated on the testing set. We set the RBF kernel parameter $\gamma = \{2^{-5}\gamma_0, 2^{-4}\gamma_0, \ldots, 2^5\gamma_0\}$, where γ_0 is the average of pairwise sample distances, the regularization parameter $C = \{10^{-3}, 10^{-2}, \ldots, 10^3\}$. Altogether, there are 77 different parameter combinations. The optimal parameter λ in LLSVM^2 is chosen from $\{10^{-5}, 10^{-4}, \ldots, 10^5\}$ based on validation. The parameter k (i.e., the number of landmark points) is fixed to 100 and a in (17) is fixed to 0.2 for all data sets.

In Fig. 2, we report the prediction accuracies of LLSVM and LLSVM^2 as the x and y coordinates in a 2-D scatterplot, under the 77 different choices of the hyperparameters. The accuracies of LLSVM^2 are based on the average of five repetitions for each parameter choice, and points above the $y = x$ line indicate an accuracy improvement of LLSVM^2 (y-axis) over LLSVM (x-axis). As can be seen, a substantial fraction of the accuracies is improved by using $\{\mathbf{X}_{\mathscr{L}}, \mathbf{y}_{\mathscr{L}}\}$ to update the kernel map. The improvement is quite significant, in case, inappropriate parameters are used when training the model, indicating the value of LLSVM^2. We can observe that in all the data sets, the best performance of LLSVM and LLSVM^2 are similar (the point closest to the top right corner always lie on the $y = x$ line); however, when the hyperparameters change among a wide range of choices, we can observe that averagely LLSVM^2 outperforms LLSVM. In other words, the resultant classifier will be much more stable with respect to the choice of hyperparameters, which is a useful property in practical classification tasks.

We also studied the performance of LLSVM^2 over the number of labeled samples, $|\mathscr{L}|$, used in learning the kernel spectrum. Empirically, we find that the performance is not sensitive to the number of labeled samples (e.g., $|\mathscr{L}| = \{100, 200, 800\}$). Due to space limit, we will omit the results.

Our experimental results show that using the class label information in finding the low-rank feature representation, the classification accuracy will be more stable and less dependent on the choice of kernel parameters. This can greatly alleviate the burden of choosing parameters through cross-validation, i.e., one can expect to search on fewer choices of the hyperparameters to obtain comparable performance.

2) Accuracy Comparison of Different Supervised Low-Rank Approximation Methods: In the section, we compare LLSVM^2 with two supervised low-rank approximation methods: 1) Fourier kernel learning (FKL): kernel learning on

TABLE VI
ACCURACIES OF DIFFERENT ALGORITHMS

Datasets	LLSVM	RKS	Fourier Kernel Learning	Nyström Kernel Learning	LLSVM^2
DIGIT	97.76±0.09	93.16±1.83	94.80±0.78	97.86±0.83	**98.36±0.09**
COIL2	89.48±0.41	89.72±1.82	90.07±2.14	**93.84±0.86**	93.20±0.02
SVMGUIDE	96.13±0.01	95.90±0.11	93.99±1.05	96.31±0.30	**96.90±0.03**
GISETTE	96.56±0.11	84.02±1.12	92.06±0.46	***96.92±0.63***	**97.40±0.02**
USPS	92.82±0.19	88.90±0.61	89.35±0.63	93.54±0.51	**94.10±0.06**
PENDIGIT	98.27±0.11	97.23±0.36	95.67±1.39	***98.30±0.95***	**98.60±0.01**

its Fourier transform [45] and 2) Nyström kernel learning (NKL): Nystöm low-rank kernel decomposition with supervised information [38]. We include LLSVM and RKS as two baseline methods. Data sets are summarized in Table V. We use the same parameter setting as in Section VI-B1 for LLSVM and LLSVM^2. The kernel parameter γ and regularizer parameter C for RKS and Nysöm Kernel Learning are the same as LLSVM^2. The number of features in the approximated kernel space is fixed to 100 for all algorithms. The λ parameter in LLSVM^2 is chosen from $\{10^{-5}, 10^{-4}, \ldots, 10^{5}\}$. For LLSVM and RKS, the optimal parameter combination is chosen by five-folds cross validation. For supervised low-rank approximation algorithms (FKL, NKL, and LLSVM^2), we randomly choose 100 labeled samples to improve the kernel map and use the rest training data as validation set. The prediction accuracy is evaluated on test set. Table VI reports evaluation results based on 10 repetitions. The best accuracy is in bold, while the accuracies that were not significantly worse (with $p > 0.05$ using the one-sided t-test) are in bold and italic. For the two baselines, LLSVM is almost always better than RKS (except in data set COIL2), which is consistent with our results in Section VI-A. FKL may result in poor local optima due to nonconvex objective function. Both NKL and LLSVM^2 have better results than LLSVM. Overall, LLSVM^2 is among the best.

VII. Conclusion

In this paper, we proposed a new approach to linearize kernel SVM for large-scale classification via low-rank decomposition of kernel matrix. This allows us to fully exploit algorithmic advances in linear SVM optimization and low-rank approximation in building highly efficient nonlinear SVM solvers for large-scale data using limited computing resources. We also proposed a metric learning approach to incorporate class label in computing the low-rank approximation, and experimental results demonstrate both the efficiency and effectiveness of the proposed algorithms. In the future, we will apply numerical perturbation analysis to provide tighter bounds on the solution of the QP problem with low-rank kernels. It would be interesting to improve the scalability by parallelization.

References

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

[2] R. Mall and J. A. K. Suykens, "Very sparse LSSVM reductions for large-scale data," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 26, no. 5, pp. 1086–1097, May 2015.

[3] B. Gu, V. S. Sheng, K. Y. Tay, W. Romano, and S. Li, "Incremental support vector learning for ordinal regression," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 26, no. 7, pp. 1403–1416, Jul. 2015.

[4] J. C. Platt, "Fast training of support vector machines using sequential minimal optimization," in *Advances in Kernel Methods*. Cambridge, MA, USA: MIT Press, 1999, pp. 185–208.

[5] C.-C. Chang and C.-J. Lin, "LIBSVM: A library for support vector machines," *ACM Trans. Intell. Syst. Technol.*, vol. 2, no. 3, pp. 27:1–27:27, 2011. [Online]. Available: http://www.csie.ntu.edu.tw/~cjlin/libsvm

[6] I. W. Tsang, J. T. Kwok, and P.-M. Cheung, "Core vector machines: Fast SVM training on very large data sets," *J. Mach. Learn. Res.*, vol. 6, pp. 363–392, Apr. 2005.

[7] A. Bordes, S. Ertekin, J. Weston, and L. Bottou, "Fast kernel classifiers with Online and active learning," *J. Mach. Learn. Res.*, vol. 6, pp. 1579–1619, Oct. 2005.

[8] Z. A. Zhu, W. Chen, G. Wang, C. Zhu, and Z. Chen, "P-packSVM: Parallel primal gradient descent kernel SVM," in *Proc. 9th IEEE Int. Conf. Data Mining*, Dec. 2009, pp. 677–686.

[9] T. Joachims and C.-N. J. Yu, "Sparse kernel SVMs via cutting-plane training," *Mach. Learn.*, vol. 76, pp. 179–193, Sep. 2009.

[10] C.-J. Hsieh, S. Si, and I. S. Dhillon, "A divide-and-conquer solver for kernel support vector machines," in *Proc. 31st Int. Conf. Mach. Learn.*, Jan. 2014, pp. 566–574.

[11] Z. Wang, K. Crammer, and S. Vucetic, "Breaking the curse of kernelization: Budgeted stochastic gradient descent for large-scale SVM training," *J. Mach. Learn. Res.*, vol. 13, pp. 3103–3131, Oct. 2012.

[12] T. Zhang, "Solving large scale linear prediction problems using stochastic gradient descent algorithms," in *Proc. 21st Int. Conf. Mach. Learn.*, 2004, p. 116.

[13] S. Shalev-Shwartz, Y. Singer, and N. Srebro, "Pegasos: Primal estimated sub-gradient solver for SVM," in *Proc. 24th Int. Conf. Mach. Learn.*, 2007, pp. 807–814.

[14] T. Joachims, "Training linear SVMs in linear time," in *Proc. 12th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2006, pp. 217–226.

[15] C.-J. Hsieh, K.-W. Chang, C.-J. Lin, S. S. Keerthi, and S. Sundararajan, "A dual coordinate descent method for large-scale linear SVM," in *Proc. 25th Int. Conf. Mach. Learn*, 2008, pp. 408–415.

[16] H.-F. Yu, C.-J. Hsieh, K.-W. Chang, and C.-J. Lin, "Large linear classification when data cannot fit in memory," *ACM Trans. Knowl. Discovery Data*, vol. 5, p. 23, Feb. 2012.

[17] Y.-W. Chang, C.-J. Hsieh, K.-W. Chang, M. Ringgaard, and C.-J. Lin, "Training and testing low-degree polynomial data mappings via linear SVM," *J. Mach. Learn. Res.*, vol. 11, pp. 1471–1490, Jan. 2010.

[18] S. Sonnenburg and V. Franc, "COFFIN: A computational framework for linear SVMs," in *Proc. 27th Int. Conf. Mach. Learn.*, 2010, pp. 999–1006.

[19] A. Rahimi and B. Recht, "Random features for large-scale kernel machines," in *Proc. Adv. NIPS*, 2007, pp. 1177–1184.

[20] S. Xiao, M. Tan, D. Xu, and Z. Y. Dong, "Robust kernel low-rank representation," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 27, no. 11, pp. 2268–2281, Nov. 2016.

[21] T. Yang, Y.-F. Li, M. Mahdavi, R. Jin, and Z.-H. Zhou, "Nyström method vs random Fourier features: A theoretical and empirical comparison," in *Proc. Adv. NIPS*, 2012, pp. 476–484.

[22] C. K. I. Williams and M. Seeger, "Using the Nyström method to speed up kernel machines," in *Proc. Adv. NIPS*, 2001, pp. 682–688.

[23] K. Zhang, I. W. Tsang, and J. T. Kwok, "Improved Nyström low-rank approximation and error analysis," in *Proc. 25th Int. Conf. Mach. Learn*, 2008, pp. 1232–1239.

[24] K. Zhang and J. T. Kwok, "Clustered Nyström method for large scale manifold learning and dimension reduction," *IEEE Trans. Neural Netw.*, vol. 21, no. 10, pp. 1576–1587, Oct. 2010.

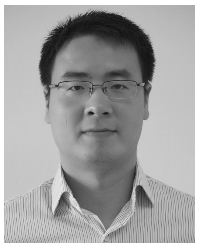
[25] C. Cortes, M. Mohri, and A. Talwalkar, "On the impact of kernel approximation on learning accuracy," in *Proc. Int. Conf. Artif. Intell. Statist.*, 2010, pp. 113–120.

[26] K. Zhang, C. Liu, J. Zhang, H. Xiong, E. Xing, and J. Ye, "Randomization or condensation?: Linear-cost matrix sketching via cascaded compression sampling," in *Proc. 23rd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2017, pp. 615–623.

[27] P. Drineas and M. W. Mahoney, "On the Nyström method for approximating a Gram matrix for improved kernel-based learning," *J. Mach. Learn. Res.*, vol. 6, pp. 2153–2175, Dec. 2005.

[28] K.-M. Lin and C.-J. Lin, "A study on reduced support vector machines," *IEEE Trans. Neural Netw.*, vol. 14, no. 6, pp. 1449–1459, Nov. 2003.

[29] K.-W. Chang and D. Roth, "Selective block minimization for faster convergence of limited memory large-scale linear models," in *Proc. 17th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2011, pp. 699–707.
[30] K. Zhang, L. Lan, Z. Wang, and F. Mörchen, "Scaling up kernel SVM on limited resources: A low-rank linearization approach," in *Proc. Int. Conf. Artif. Intell. Statist.*, 2012, pp. 1425–1434.
[31] S. Kumar, M. Mohri, and A. Talwalkar, "Sampling techniques for the Nyström method," in *Proc. 12th Int. Conf. Artif. Intell. Statist.*, 2009, pp. 304–311.
[32] F. Bach, "Sharp analysis of low-rank kernel matrix approximations," in *Proc. Conf. Learn. Theory*, 2013, pp. 185–209.
[33] E. P. Xing, A. Y. Ng, M. I. Jordan, and S. Russell, "Distance metric learning, with application to clustering with side-information," in *Proc. Adv. NIPS*, 2003, pp. 521–528.
[34] C. Fowlkes, S. Belongie, F. Chung, and J. Malik, "Spectral grouping using the Nyström method," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 26, no. 2, pp. 214–225, Feb. 2004.
[35] Y.-J. Lee and O. L. Mangasarian, "RSVM: Reduced support vector machines," in *Proc. 1st SIAM Int. Conf. Data Mining*, 2001, pp. 1–17.
[36] M. Wu, B. Schölkopf, and G. Bakır, "A direct method for building sparse kernel learning algorithms," *J. Mach. Learn. Res.*, vol. 7, pp. 603–624, Apr. 2006.
[37] F. R. Bach and M. I. Jordan, "Predictive low-rank decomposition for kernel methods," in *Proc. 22nd Int. Conf. Mach. Learn*, 2005, pp. 33–40.
[38] K. Zhang, L. Lan, J. Liu, A. Rauber, and F. Moerchen, "Inductive kernel low-rank decomposition with priors: A generalized Nyström method," in *Proc. 29th Int. Conf. Mach. Learn*, 2012, pp. 305–312.
[39] G. R. G. Lanckriet, N. Cristianini, P. Bartlett, L. El Ghaoui, and M. I. Jordan, "Learning the kernel matrix with semidefinite programming," *J. Mach. Learn. Res.*, vol. 5, pp. 27–72, Jan. 2004.
[40] B. Kulis, M. A. Sustik, and I. S. Dhillon, "Low-rank kernel learning with Bregman matrix divergences," *J. Mach. Learn. Res.*, vol. 10, pp. 341–376, Feb. 2009.
[41] Q. Mao, I. W. Tsang, S. Gao, and L. Wang, "Generalized multiple kernel learning with data-dependent priors," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 26, no. 6, pp. 1134–1148, Jun. 2015.
[42] Z. Wang, N. Djuric, K. Crammer, and S. Vucetic, "Trading representability for scalability: Adaptive multi-hyperplane machine for nonlinear classification," in *Proc. 17th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2011, pp. 24–32.
[43] N. Djuric, L. Lan, S. Vucetic, and Z. Wang, "BudgetedSVM: A toolbox for scalable SVM approximations," *J. Mach. Learn. Res.*, vol. 14, pp. 3813–3817, Jan. 2013.
[44] Q. Le, T. Sarlós, and A. Smola, "Fastfood—Computing Hilbert space expansions in loglinear time," in *Proc. 30th Int. Conf. Mach. Learn*, 2013, pp. 244–252.
[45] E. G. Băzăvan, F. Li, and C. Sminchisescu, "Fourier kernel learning," in *Proc. Eur. Conf. Comput. Vis.*, 2012, pp. 459–473.

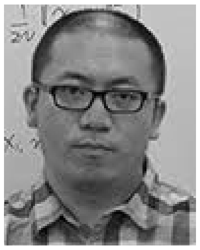


Liang Lan received the B.S. degree in bioinformatics from the Huazhong University of Science and Technology, Wuhan, China, in 2007, and the Ph.D. degree in computer and information sciences from Temple University, Philadelphia, PA, USA, in 2012.

He is currently an Assistant Professor with the Department of Computer Science, Hong Kong Baptist University, Hong Kong. His current research interests include data mining and machine learning.

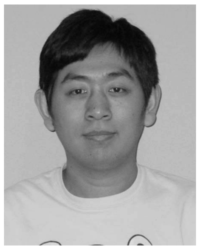
Zhuang Wang received the B.A. degree in electronic commerce from Wuhan University, Wuhan, China, in 2006, and the Ph.D. degree in computer and information sciences from Temple University, Philadelphia, PA, USA, in 2010.

From 2010 to 2013, he was with Siemens Corporate Research, Princeton, NJ, USA. From 2013 to 2014, he was with IBM Global Business Services, Philadelphia, PA, USA. From 2014 to 2015, he was with Skytree, San Francisco, CA, USA. He is currently a Research Scientist with Facebook, Menlo Park, CA, USA. His current research interests include machine learning and data mining.



Shandian Zhe received the Ph.D. degree in computer science from Purdue University, West Lafayette, IN, USA, in 2017.

He is currently an Assistant Professor with the School of Computing, University of Utah, Salt Lake City, UT, USA. His current research interests include machine learning and data science, including Bayesian nonparametrics, approximate inference, Bayesian deep learning, large-scale machine learning, tensor/matrix factorization, computational biology, and computational advertisement.



Wei Cheng received the Ph.D. degree from the University of North Carolina at Chapel Hill, Chapel Hill, NC, USA, in 2015.

He is currently a Research Staff Member with the Data Science Department, NEC Laboratories Americ, Princeton, NJ, USA. His current research interests include data mining, machine learning, and bioinformatics.

Jun Wang received the Ph.D. degree in electrical engineering from Columbia University, New York, NY, USA, in 2011.

From 2010 to 2014, he was a Research Staff Member with the IBM T. J. Watson Research Center, Yorktown Heights, NY, USA. He is with the Institute of Data Science and Technology, Alibaba Group, Hangzhou, China. He is currently a Professor with the School of Computer Science and Software Engineering, East China Normal University, Shanghai, China, and an Adjunct Faculty Member with Columbia University, New York, NY, USA. His current research interests include machine learning, data mining, mobile intelligence, and computer vision.

Dr. Wang was a recipient of the Thousand Talents Plan in 2014, the Outstanding Technical Achievement Award from IBM in 2013, and the Jury Thesis Award from Columbia University.

Kai Zhang received the Ph.D. degree in computer science from the Hong Kong University of Science and Technology, Hong Kong, in 2008.

He joined the Lawrence Berkeley National Laboratory, Berkeley, CA, USA. He is currently an Associate Professor with the Department of Computer and Information Sciences, Temple University, Philadelphia, PA, USA. His current research interests include randomized algorithms for matrix decomposition, large-scale machine learning algorithms, and applications in bioinformatics, time series modeling, and brain functional networks.

Dr. Zhang was a recipient of the 2016 ACM SIGKDD Best Paper Runner-up Award and the 2016 Business Contribution Award of NEC Laboratories America in Princeton.