Artificial Intelligence 250 (2017) 1–15

Contents lists available at ScienceDirect

Artificial Intelligence

www.elsevier.com/locate/artint

Low-rank decomposition meets kernel learning: A generalized Nyström method

Liang Lan [a], Kai Zhang [b,*], Hancheng Ge [c], Wei Cheng [d], Jun Liu [e], Andreas Rauber [f], Xiao-Li Li [g], Jun Wang [h], Hongyuan Zha [i,h]

[a] Lenovo Group Limited, 100 Cyberport Road, Hong Kong
[b] Department of Computer and Information Sciences, Temple University, Philadelphia PA 19122, United States
[c] Department of Computer Science and Engineering, Texas A&M University, Texas 77840, United States
[d] NEC Laboratories America, Princeton 08540, United States
[e] SAS Institute Inc., Cary, NC 27513, United States
[f] Institute of Softwares and Interactive Systems, Technical University of Vienna, Austria
[g] Institute for Infocomm Research, 138632, Singapore
[h] School of Computer Science and Software Engineering, East China Normal University, Shanghai 2000062, China
[i] College of Computing, Georgia Institute of Technology, Atlanta, GA 30332, United States

ARTICLE INFO

Article history:
Received 2 March 2016
Received in revised form 30 April 2017
Accepted 9 May 2017
Available online 15 May 2017

Keywords:
Kernel learning
Nyström low-rank decomposition
Large-scale learning algorithms
Multiple kernel learning

ABSTRACT

Low-rank matrix decomposition and kernel learning are two useful techniques in building advanced learning systems. Low-rank decomposition can greatly reduce the computational cost of manipulating large kernel matrices. However, existing approaches are mostly unsupervised and do not incorporate side information such as class labels, making the decomposition less effective for a specific learning task. On the other hand, kernel learning techniques aim at constructing kernel matrices whose structure is well aligned with the learning target, which improves the generalization performance of kernel methods. However, most kernel learning approaches are computationally very expensive. To obtain the advantages of both techniques and address their limitations, in this paper we propose a novel kernel low-rank decomposition formulation called the generalized Nyström method. Our approach inherits the linear time and space complexity via matrix decomposition, while at the same time fully exploits (partial) label information in computing task-dependent decomposition. In addition, the resultant low-rank factors can generalize to arbitrary new samples, rendering great flexibility in inductive learning scenarios. We further extend the algorithm to a multiple kernel learning setup. The experimental results on semi-supervised classification demonstrate the usefulness of the proposed method.

1. Introduction

Advanced learning systems typically have two basic requirements: *computational efficiency* and *generalization ability*. In this paper, we are interested in building kernel based learning algorithms. To generate desired kernel machines catering to these requirements, significant research efforts have been devoted to low-rank matrix decomposition and kernel learning.

* Corresponding author.
E-mail addresses: llan@lenovo.com (L. Lan), kzhang980@gmail.com (K. Zhang), hge@cse.tamu.edu (H. Ge), wcheng@nec-labs.com (W. Cheng), jun.liu@sas.com (J. Liu), rauber@ifs.tuwien.ac.at (A. Rauber), xlli@i2r.a-star.edu.sg (X.-L. Li), wongjun@gmail.com (J. Wang), zha@cc.gatech.edu (H. Zha).

http://dx.doi.org/10.1016/j.artint.2017.05.001
0004-3702/

Low-rank decomposition aims at factorizing a Positive Semi-Definite (PSD) kernel matrix into a product of low-rank factorsp. Given an $n \times n$ kernel matrix K with rank $r << n$, the kernel matrix can be represented as $K = GG^\top$, with G an $n \times r$ matrix. Low-rankness is an important property that has been widely exploited in machine learning. For example, the kernel matrix often has a rapidly decaying spectrum and thus small rank, and eigenvectors corresponding to large eigenvalues create useful basis functions for classification [1,2]. The low-rank property is also quite useful in reducing the memory and computational cost in large scale problems using low-rank approximations. It has been used widely to scale up various machine learning algorithms [3–7]. However, there are certain limitations with existing approaches. First, many such methods are intrinsically unsupervised and only focus on numerical approximation of the kernel matrix. For example, well known decomposition techniques such as eigenvalue decomposition, QR factorization, Cholesky decomposition are all unsupervised methods. When confronted with a specific learning task, however, we believe that incorporating side information (such as partially labeled samples or grouping constraints) will lead to more effective decomposition and improved performance. Second, many existing decomposition methods, such as QR decomposition, incomplete Cholesky factorization [8], work in a transductive manner. That means the factorization can only be computed for samples that are available in the training stage. For new samples that are only available in the testing stage, it is not straightforward to generalize the decomposition to these data, and the difficulty becomes more pronounced if (partial) label information is considered.

Apart from computational efficiency, kernel learning is another research area that aims to improve the generalization performance of kernel methods. Note that kernel based learning algorithms implicitly embed data in a Euclidean space [9] via the kernel matrix. The goal of kernel learning therefore is to obtain a kernel matrix such that the induced geometry (1) preserves the original similarities in the data and, (2) correlates well with the class labels and henceforth better predicts them. Lanckriet et al. [9] proposed to learn the kernel matrix whose embedding shows maximal margin while keeping the trace of the kernel matrix constant. Instead of using the margin of SVM as the optimization criterion, Cristianini et al. [10] proposed the *kernel alignment* as a general objective independent of classifiers. Maximizing alignment of the learned kernel with the *ideal kernel* [11] provides a general and effective way for kernel design. The concept of *ideal kernel* and *kernel alignment* has led to a family of Multiple Kernel Learning (MKL) algorithms [10,12,9,13–15]. The key idea of MKL is to choose a set of base kernels and then optimize their weighting coefficients via maximizing the alignment with the ideal kernel. Recently, an improved alignment criterion called centralized kernel alignment was proposed by Cortes et al. [16]. Here the kernel matrix is subject to double-centering before computing their correlations, which has been shown to overcome some of the limitations with the traditional kernel alignment and provides a novel concentration bound in both classification and regression tasks. Cortes et al. [16] showed that the weighting coefficients in the maximal alignment kernel can be solved via Quadratic Programming (QP). In [17], an order constraint on the weighting coefficients is imposed that gives higher priority to eigenvectors corresponding to smaller eigenvalues of the graph Laplacian.

Although kernel learning algorithms can improve the quality of the kernel matrix and therefore the generalization performance, the computational cost is typically demanding. For instance, most algorithms will need to store multiple $n \times n$ base kernel matrices, which takes at least $O(n^2)$ space and time. In addition, computing the eigenvalue decomposition of the kernel matrix takes $O(n^2)$ space and $O(n^3)$ time. This has become a bottleneck in many real-life applications with large scale data.

To summarize, both low-rank decomposition and kernel learning have their advantages and limitations. The former has a computational advantage but is not very well-tailored to fit specific learning tasks; the latter usually involves careful design to boost the generalization performance, but the high computational cost makes it impractical for big data problems.

To obtain the advantages of both low-rank decomposition and kernel learning, in this paper we propose a novel low-rank decomposition algorithm that incorporates side information in producing efficient and effective low-rank kernel. We achieve this by generalizing the Nyström method in a novel way. The Nyström method is a sampling based approach and has gained great popularity in unsupervised kernel low-rank approximation, with both theoretical performance guarantees and empirical successes [5,6,18]. Our main novelty is to provide an interesting interpretation of the matrix completion view of the Nyström method as a an extrapolation of a dictionary kernel, and generalize it to incorporate side information in computing improved low-rank decompositions. Our approach has two important advantages. First, it has a flexible, generative structure that allows us to generalize computed low-rank factorizations to arbitrary new unseen samples. Second, both the space and time complexities of our approach are linear in the sample size, rendering great efficiency in learning a useful low-rank kernel.

Our method also has connections to sparse Gaussian process [19–24]. An important goal of sparse Gaussian process is to determine the choice of a small set of "pseudo"-samples to build a sparse model with significantly reduced computational cost. This shows the similarity to sample based approximations of the kernel matrix. Our approach shares some features with sparse GP but has some distinct aspects. We provide an in-depth discussion in Section 5.

The rest of the paper is organized as follows. In Section 2, we introduce the Nyström method. In Section 3, we propose the generalized Nyström low-rank decomposition incorporating side information. In Section 4 we discuss the multiple kernel version of the generalized Nyström method. In Section 5 we discuss related work. In Section 6 we report empirical evaluation results. The last section concludes the paper. A preliminary conference version of this paper is published by Zhang et al. [25].

2. Nyström method

The Nyström method is a sampling based algorithm for approximating large kernel matrices and their eigen-systems. It originated from solving integral equations and was introduced to the machine learning community by Williams and Seeger [5], Fowlkes et al. [6].

Given a kernel function $k(\cdot,\cdot)$ and a sample set with underlying distribution $p(\cdot)$, the Nyström method aims at solving the following integral equation

$$\int k(\mathbf{x},\mathbf{y})p(\mathbf{y})\phi_i(\mathbf{y})d\mathbf{y} = \lambda_i\phi_i(\mathbf{x}). \tag{1}$$

Here $\phi_i(\mathbf{x})$ and λ_i are the ith eigenfunction and eigenvalue of the operator $k(\cdot,\cdot)$ with regard to p. By drawing a set of m samples $\mathcal{Z}$ (landmark points) from $p(\cdot)$, an empirical average can be computed as

$$\frac{1}{m}\sum_{j=1}^{m} k(\mathbf{x},\mathbf{z}_j)\phi_i(\mathbf{z}_j) = \lambda_i\phi_i(\mathbf{x}) \tag{2}$$

By choosing $\mathbf{x}$ in (2) as $\mathbf{z}_1, \mathbf{z}_2, ..., \mathbf{z}_m$, one has an eigenvalue decomposition as $W\phi_i = \lambda_i\phi_i$, where $W \in \mathbf{R}^{m\times m}$ is the kernel matrix defined on landmark points, $\phi_i \in \mathbf{R}^{m\times 1}$ and λ_i are the ith eigenvector and eigenvalue of W. Note that the Nyström method is a special case of collocation method [26], which approximates the eigenfunctions $\phi_i(\mathbf{x})$ by a linear combination of a set of basis functions and solves (1) based on selected collocation points.

In practice, given a large data set $\mathcal{X} = \{\mathbf{x}_i\}_{i=1}^n$, the Nyström method randomly selects m landmark points $\mathcal{Z}$ with $m \ll n$, and computes the eigenvalue decomposition of W. Then the eigenvectors of W are extrapolated to the whole sample set by (2). Interestingly, the Nyström method is shown to implicitly reconstruct the whole $n \times n$ kernel matrix K by

$$K \approx EW^{\dagger}E^{\top}. \tag{3}$$

Here $W^{\dagger}$ is the pseudo-inverse of W, and $E \in \mathbf{R}^{n\times m}$ is the kernel matrix defined on the sample set $\mathcal{X}$ and landmark points $\mathcal{Z}$. The Nyström method requires $O(mn)$ space and $O(m^2n)$ time, which are linear in the sample size considering $m \ll n$. It has drawn considerable interest in clustering and manifold learning [27,18], Gaussian processes [5], and kernel methods [3].

Landmark selection in Nyström method. Selection of the landmark points $\mathcal{Z}$ in the Nyström method can have a great impact on its performance. In the literature, various sampling schemes have been proposed. The simplest approach is random sampling [5,6]. Recently various non-uniform sampling schemes are also studied. Smola and Schökopf [28] proposed a greedy sampling scheme to sequentially select columns which maximally decreases the approximation error; Kumar et al. [29] proposed an adaptive sampling scheme that only requires visiting part of the kernel matrix, together with relative error bound; Drineas and Mahoney [30] proposed a probabilistic sampling scheme where the probability for choosing the ith row/column of the kernel matrix K is proportional to the norm of that row/column.

Zhang et al. [7], Zhang and Kwok [18] proposed to select the landmark points as the k-means clustering centers of the input data. They showed that the Nyström low-rank approximation error is closely related to the error of reconstructing input samples with selected landmark points. In previous evaluations [31], researchers have found that k-means based sampling method outperforms many other sampling schemes (random, greedy, or adaptive) by a clear margin, and at the same time being computationally very efficient.

In this work we adopt the k-means based sampling because of its efficiency and competitive performance. Other sampling schemes are applicable as well, but our main focus is on how to generalize the procedures of the Nyström method such that it can further incorporate class label information.

3. Generalized Nyström low-rank decomposition

3.1. Extrapolation of the dictionary kernel

We first present an interesting interpretation of the matrix completion view of the Nyström method (3). It reconstructs (i, j)th entry of the kernel matrix as

$$K_{ij} = E_i W^{\dagger} E_j^{\top}, \tag{4}$$

where $E_i \in \mathbf{R}^{1\times m}$ is the ith row of the extrapolation matrix E. In other words, the similarity between any two samples $\mathbf{x}_i$ and $\mathbf{x}_j$ is constructed by first computing their respective similarities to the landmark set (E_i and E_j), and then modulated by the inverse of the similarities among the landmark points $W^{\dagger}$. With regards to this, we have the following proposition.

Proposition 1. *Given m landmark points $\mathcal{Z}$, we use (4) to construct the similarity between any two samples, $\mathbf{x}_i$ and $\mathbf{x}_j$. Let $\mathbf{z}_p$ and $\mathbf{z}_q$ be the closest landmark points to $\mathbf{x}_i$ and $\mathbf{x}_j$, respectively. Let $d_p = \|\mathbf{x}_i - \mathbf{z}_p\|_2$, and $d_q = \|\mathbf{x}_j - \mathbf{z}_q\|_2$. Let the kernel function $k(\cdot,\cdot)$ satisfy*

the Lipschitz property[1] $|k(\mathbf{x},\mathbf{y})-k(\mathbf{x},\mathbf{z})| \le \eta\|\mathbf{y}-\mathbf{z}\|_2$, *and* $c = \max k(\cdot,\cdot)$. *Then the reconstructed similarity* K_{ij} *and the* (p,q)*th entry of* W *will have the following relation*

$$|K_{ij} - W_{pq}| \le \sqrt{m}\eta(cd_p + cd_q + \sqrt{m}\eta d_p d_q)\|W^\dagger\|_F.$$

Proof. Let $E_i = [k(\mathbf{x}_i,\mathbf{z}_1)\, k(\mathbf{x}_i,\mathbf{z}_2)\ \dots\ k(\mathbf{x}_i,\mathbf{z}_m)]$, and $W_p = [k(\mathbf{z}_p,\mathbf{z}_1)\, k(\mathbf{z}_p,\mathbf{z}_2)\ \dots\ k(\mathbf{z}_p,\mathbf{z}_m)\ ]$, and define $\Delta_p = E_i - W_p$, $\Delta_q = E_j - W_q$. We have $\|\Delta_p\|_2^2 = \sum_{o=1}^m (k(\mathbf{x}_i,\mathbf{z}_o) - k(\mathbf{z}_p,\mathbf{z}_o))^2 \le m\eta^2 d_p^2$, similarly, $\|\Delta_q\|_2^2 \le m\eta^2 d_q^2$. We also have $\|E_i\|_2^2, \|E_j\|_2^2 \le c$, then we have

$$\begin{aligned}
|K_{ij} - W_{pq}| &= |(W_p + \Delta_p)^\top W^\dagger (W_q + \Delta_q) - W_{pq}| \\
&= \left|(W_p^\top W^\dagger W_q - W_{pq}) + W_p^\top W^\dagger \Delta_q + \Delta_p^\top W^\dagger W_q + \Delta_p W^\dagger \Delta_q\right| \\
&\le cm(\|\Delta_p\|_2 + \|\Delta_q\|_2)\|W^\dagger\|_F + \|\Delta_p\|_2\|\Delta_q\|_2\|W^\dagger\|_F \\
&= (cm\|\Delta_p\|_2 + cm\|\Delta_q\|_2 + \|\Delta_p\|_2\|\Delta_q\|_2)\|W^\dagger\|_F \\
&= m\eta(c\sqrt{m}d_p + c\sqrt{m}d_q + \eta d_p d_q)\|W^\dagger\|_F.
\end{aligned}$$

Here we used the equality $W_p W^\dagger W_q^\top = W_{pq}$, since W_p and W_q are the pth and qth row (column) of W. □

Proposition 1 gives an interesting interpretation of the kernel reconstruction mechanism of the Nyström method (4). If $\mathbf{x}_i$ and $\mathbf{x}_j$ happen to coincide with a pair of landmark points, $\mathbf{z}_p$ and $\mathbf{z}_q$, then $K_{ij} = W_{pq}$, i.e., the pqth entry of W will be extrapolated exactly to $(\mathbf{x}_i,\mathbf{x}_j)$. In case $\mathbf{x}_i$ and $\mathbf{x}_j$ do not coincide with any landmark point, the difference between K_{ij} and W_{pq}, with $\mathbf{z}_p$ and $\mathbf{z}_q$ being the closest landmark points to $\mathbf{x}_i$ and $\mathbf{x}_j$, will be bounded by the distances $\|\mathbf{x}_i - \mathbf{z}_p\|_2$ and $\|\mathbf{x}_j - \mathbf{z}_q\|_2$. The smaller the distances, the closer K_{ij} and W_{pq}. In other words, the similarity matrix W on the landmark points serves as a *dictionary kernel*, whose entries are extrapolated (on both sides) to any pairs of samples $(\mathbf{x}_i,\mathbf{x}_j)$ according to the proximity relation between landmark points and samples, and the reconstruction is exact on the landmark points $\mathcal{Z}$ which serve as the "nodes" for extrapolation.

3.2. Including side information

The kernel extrapolation view of the Nyström method (Proposition 1) inspires us to generalize it to handle side information in learning a low-rank kernel. Note that quality of the *dictionary kernel* will have a large impact on the whole kernel matrix. In the original Nyström method (3), the dictionary kernel W is simply computed as the pairwise similarity between landmark points, which can deviate from an "ideal" one. Therefore, instead of using such an "unsupervised" dictionary kernel, we propose to learn a new dictionary kernel that better coincides with given side information.

Suppose we are given a set of labeled and unlabeled samples. Let $\mathcal{Z}$ be a set of m pre-selected landmark points. Let $E \in \mathbf{R}^{n\times m}$ be the extrapolation kernel matrix between samples $\mathcal{X}$ and landmark $\mathcal{Z}$, and let $E_l \in \mathbf{R}^{l\times m}$ be the rows of E corresponding to labeled samples. For simplicity, let $S_0 = W^\dagger$ denote the inverse of the dictionary kernel in the standard Nyström method (3). Our task is to learn (the inverse of) a new dictionary kernel, denoted by S, subject to the following considerations:

1. **unsupervised information**: the reconstructed kernel $ESE^\top$ should preserve the structure of the original kernel matrix K, since K encodes important pairwise relation between samples;
2. **supervised information**: the reconstructed kernel on the labeled samples, $E_l S E_l^\top$, should be consistent with the given side information.

Note that in the standard Nyström method, $EW^\dagger E^\top$ provides an effective approximation of K. Therefore, to achieve the first goal, we use

$$S_0 = \beta \cdot W^\dagger \tag{5}$$

as a prior for the (inverse) dictionary kernel S, namely, they should be close to each other. The choice of the scaling factor β will be clear in the sequel.

To achieve the second goal, we use the concept of kernel target alignment [10] and require that the reconstructed kernel, $E_l S E_l^\top$, is close to the ideal kernel K_l^* defined on labeled samples. The ideal kernel is defined as [11]

$$[K_l^*]_{ij} = \begin{cases} 1 & \mathbf{x}_i, \mathbf{x}_j \text{ in the same class} \\ 0 & \text{otherwise.} \end{cases} \tag{6}$$

[1] Previous work [18] shows that both stationary kernels $k(x,y) = \kappa(\|x-y\|/\sigma)$ and polynomial kernels $k(x,y) = (x'y+\epsilon)^d$ satisfy the Lipschitz property. Since Gaussian kernel is shift-invariant, it also satisfies the Lipschitz condition.

We therefore arrive at the following problem

$$\min_{S \in R^{m \times m}} \lambda \|S - S_0\|_F^2 + \|E_I S E_I^\top - K_I^*\|_F^2 \quad s.t.\ S \succeq 0. \tag{7}$$

Here, we used the Frobenius norm to measure the closeness between two matrices. Note that in [10], the closeness between two kernel matrices is measured by their inner product $\langle K_1, K_2 \rangle = \sum_{ij} K_1(\mathbf{x}_i, \mathbf{x}_j) K_2(\mathbf{x}_i, \mathbf{x}_j)$. Since $\|K_1 - K_2\|_F^2 = \langle K_1, K_1 \rangle + \langle K_2, K_2 \rangle - 2\langle K_1, K_2 \rangle$, minimizing the Frobenius norm is related to maximizing the alignment. We choose the Frobenius norm here because it leads to a convex optimization, which can be computationally more feasible than if we choose kernel alignment score in [10]. We will use the normalized kernel alignment score [16] afterwards as an independent measure to choose the hyper-parameter λ. Details will be discussed in Section 3.8.

In practical implementations, we choose the scaling factor β in (5) as

$$\beta = \frac{\|E_I^\dagger K_I^* (E_I^\top)^\dagger\|_F}{\|W^\dagger\|_F} \tag{8}$$

such that solutions from the two competing terms respectively in (7), namely $W^\dagger$ and $E_I^\dagger K_I^* (E_I^T)^\dagger$, would have the same magnitude. In this case the resultant combined objective would be more meaningful. One can also optimize the scaling factor β iteratively with the other variable S, which we found in our experiments produce similar result with (8) with higher computational cost (see Appendix A). So we simply adopt (8) in our evaluations.

We call our method generalized Nyström low-rank decomposition. The method has several desirable properties. First, as long as the inverse dictionary kernel S is PSD, the resultant kernel $ESE^\top$ will also be PSD; second, the rank of the kernel matrix can be easily controlled by the landmark size; this can be computationally much more efficient than learning a full kernel matrix subject to rank constraint; third, the extrapolation (4) is "generative" and allows us to compute the similarity between any pair of samples; this means the learned kernel matrix generalizes easily to new unseen samples. Since the dictionary kernel is learned with side information, the generalization to new samples naturally incorporates such information, which provides much more convenience in updated environments.

3.3. Must-link grouping constraints as side information

Suppose we are given a set of must-link grouping constraints denoted by $\mathcal{I}$. Let $\mathcal{X}_I$ be the subset of samples with such constraints. Then we define $\mathbf{T} \in \mathbf{R}^{|\mathcal{X}_I| \times |\mathcal{X}_I|}$ such that

$$\mathbf{T}_{ij} = \begin{cases} 1 & (\mathbf{x}_i, \mathbf{x}_j) \in \mathcal{X}_I \\ 0 & \text{otherwise.} \end{cases}$$

Then our objective can be written conveniently as

$$\min_{S \in R^{m \times m}} \lambda \|S - S_0\|_F^2 + \|\mathbf{T} \odot (E_I S E_I^\top - K_I^*)\|_F^2 \quad s.t.\ S \succeq 0.$$

Here $\odot$ denotes the element-wise multiplication and K_I^* is defined similarly as in (6). The objective can be transferred to (7) by element-wise multiplying $\mathbf{T}$ with $E_I S E_I^T$ and K_I^* respectively.

Our current formulation does not handle "do-not-link" constraints directly. To achieve this, different loss functions are needed that take into account geometric relations between samples of different classes such as in [32].

3.4. Optimization

The objective (7) is convex with respect to S, and the PSD constraint $S \succeq 0$ is also convex. Therefore a global optimum can be guaranteed.

Note that S is a matrix with only m^2 variables, where $m \ll n$ is a user defined value. Therefore the problem (7) is relatively easy to optimize. We use the gradient mapping strategy [33] that is composed of iterative gradient descent equipped with a projection step to find the optimal solution. Given an initial solution $S^{(t)}$, we update it by

$$S^{(t+1)} = S^{(t)} + \eta^{(t)} \nabla_{S^{(t)}}, \tag{9}$$

where ∇_S is the gradient of the objective J (7) at S,

$$\nabla_S = 2\lambda(S - S_0) + 2E_I^\top (E_I S E_I^\top - K_I^*) E_I.$$

The step length $\eta^{(t)}$ is determined by the Armijo–Goldstein rule [33]. In particular, we start from an initial small scalar a and solve

$$B_a^* = \arg\min_{B \succeq 0} tr\big(\nabla_{S^{(t)}} B\big) + \frac{a}{2} \|B - S^{(t)}\|_F^2. \tag{10}$$

This is a standard matrix nearness problem with PSD constraint, and B_a^* can be computed in a closed-form as $S^{(t)} - \frac{1}{a}\nabla_{S^{(t)}}$ with removal of negative eigenvectors/values. Then we examine [33]

$$J(B_a^*) \leq J(S^{(t)}) + tr\Big(\nabla_{S^{(t)}}(B_a^* - S^{(t)})\Big) + \frac{a}{2}\|B_a^* - S^{(t)}\|_F^2.$$

If this inequality is violated, we increase a by a constant factor and re-calculate (10) until the relation holds. Then we use $\eta^{(t)} = \frac{1}{a}$ as the step length for (9).

After the descent step (9), we project the $S^{(t+1)}$ onto the set of positive semi-definite cones as follows

$$S^{(t+1)} \leftarrow U^{(t+1)}\Lambda_+^{(t+1)}(U^{(t+1)})^\top,$$

where $U^{(t+1)}$ and $\Lambda^{(t+1)}$ are the eigenvectors and eigenvalues of $S^{(t+1)}$, and

$$\Lambda_+^{(t+1)}{}_{ii} = \begin{cases} \Lambda_{ii}^{(t+1)} & \text{if } \Lambda_{ii}^{(t+1)} \geq 0; \\ 0 & \text{otherwise.} \end{cases} \tag{11}$$

One can also use more advanced approaches such as the Nesterov's method [33] to improve the convergence rate. We do not explore more options here because the size of our optimization problem is small and empirically it converges quickly due to a principled initialization (see next subsection).

3.5. Initialization

In this section, we propose a closed-form initialization which helps quickly locate the optimal solution. The basic idea is to drop the PSD constraint in (7) first, and compute the vanishing point of the gradient, i.e., $\frac{\partial J(S)}{\partial S} = 0$, which leads to

$$\lambda S + E_l^\top E_l S E_l^\top E_l = E_l^\top K_l^* E_l + \lambda S_0.$$

Then we have

$$S + PSP^\top = Q, \tag{12}$$

where

$$P = \frac{1}{\sqrt{\lambda}}(E_l^\top E_l),$$
$$Q = S_0 + \frac{1}{\lambda}E_l^\top K_l^* E_l.$$

Equation (12) can be solved as follows. Suppose the diagonalization of P is $P = U\Lambda U^\top$, and define $S = U\tilde{S}U^\top$, $Q = U\tilde{Q}U^\top$, then (12) reduces to

$$U\tilde{S}U^\top + U\Lambda\tilde{S}\Lambda U^\top = U\tilde{Q}U^\top \;\rightarrow\; \tilde{S} + \Lambda\tilde{S}\Lambda^\top = \tilde{Q}.$$

Since Λ is a diagonal matrix, this becomes m^2 equations

$$\tilde{S}_{ij} + \Lambda_{ii}\Lambda_{jj}\tilde{S}_{ij} = \tilde{Q}_{ij}, \;\; 1 \leq i, j \leq m.$$

Therefore we have a closed-form solution of S, as

$$S = U\tilde{S}U^\top, \tag{13}$$
$$\text{where } \tilde{S}_{ij} = \frac{\tilde{Q}_{ij}}{1 + \Lambda_{ii}\Lambda_{jj}}.$$

After computing S, we then project it onto the set of positive semi-definite cones similar to (11). Such an initial solution can be deemed as the closest PSD matrix to the unconstrained version of (7). Empirically, such an initial solution alone already leads to satisfactory prediction result.

3.6. Time and space complexity

The space complexity of our algorithm is $O(mn)$, where n is sample size and m is the number of landmark points. Computationally, it only requires repeated eigenvalue decomposition of $m \times m$ matrices, and a single multiplication between the $n \times m$ extrapolation E and the $m \times m$ dictionary kernel S. The overall time complexity is $O(m^2 n) + O(t \log(\mu_{\max}) m^3)$, where t is the number of gradient mapping iterations, and $\mu_{\max}$ is the maximum eigenvalue of the Hessian. This is because a in (10) is bounded by $\mu_{\max}$ and one can always find a suitable step-length in $\log(\mu_{\max})$ steps. Empirically, with the initialization in Section 3.5, only a few iterations are needed. Therefore t is a small integer and our algorithm has a linear time and space complexity.

3.7. Selecting the rank

The parameter m specifies the number of landmark points in the Nyström method. On the one hand, it directly determines the computational complexity, therefore the smaller the m, the more savings we can expect. On the other hand, m is closely related to the rank of the kernel matrix K. In practice, depending on the size of the kernel matrix and whether or not it really has a low-rank structure, m may be quite large. In [1], it was observed that even if the kernel matrix has a relatively high rank, using a low-rank approximation can still be useful in terms of the learning performance. Therefore in practice we typically choose m as 10% of the training sample size. In case the training data is very large and even 10% of the data still cannot fit in memory, we will set m as a small constant.

3.8. Selecting λ

The hyper-parameter λ in (7) can be difficult to choose if the side information (e.g., partially labeled samples) is limited. Here we propose a heuristic to choose λ. Note that the objective (7) contains two residuals, $S_0 - S$, and $E_l S E_l^\top - K_l^*$, in terms of the Frobenius norm, which are additive and require a tradeoff parameter λ. Here, we use a new criterion with certain invariance property to re-evaluate the goodness of the solution. More specifically, we used Normalized Kernel Alignment (NKA) [16] between kernel matrices,

$$\rho[K, K'] = \frac{\langle K_c, K'^\top_c \rangle_F}{\|K_c\|_F \|K'_c\|_F}, \tag{14}$$

where $\langle \cdot, \cdot \rangle_F$ is the Frobenius inner product, K_c is the centered kernel associated to K. The NKA score always has a magnitude that is smaller than 1. It is independent of the scale of the solution, and is multiplicative by nature. Let $S(\lambda)$ be the optimum of (7) for a fixed λ. Then we choose the best λ by

$$\lambda^* = \arg\max_{\lambda \in \mathcal{G}} \rho\left[S(\lambda), S_0\right] \cdot \rho\left[E_l S(\lambda) E_l^\top, K_l^*\right]. \tag{15}$$

Here $\mathcal{G}$ is the set of candidate λ's. The criterion (15) has the following properties: (1) it is scale invariant, and does not require any extra trade-off parameter due to its multiplicative form; (2) the first term measures the closeness between S and S_0, which is related to unsupervised structures of kernel matrix; the second term measures the closeness between $E_l S E_l^\top$ and K_l^*, which is related to side information; therefore the criterion is consistent with the objective (7) but is numerically different; (3) a higher alignment (second term in (15)) indicates existence of a good predictor with higher probability [16]; (4) computation of the criterion does not require any extra validation set, which is convenient if only limited labeled samples are available. Therefore, this is an informative criterion to measure the quality of the solution. Empirically, it correlates nicely with the prediction accuracy on the test samples, as will be reported in Section 6.1.

4. Multiple kernel low-rank decompositions

In practical scenarios, the choice of kernel parameter (such as the Gaussian kernel width) can greatly affect the performance of learning algorithms. Recent research in multiple kernel learning has emphasized the need to adopt multiple kernels, which not only provides more flexibility but also alleviates the difficulty in hyper-parameter selection.

We represent the desired kernel matrix using a mixture of low-rank decompositions. Suppose we use M different kernel parameters, each associated with a kernel matrix $K^{(j)} \in \mathbf{R}^{n \times n}$ for $j = 1, 2, \ldots, M$. Among these M kernel matrices, we use a shared landmark set $\mathcal{Z}$ and obtain their respective decomposition as

$$K^{(j)} \approx E^{(j)} S^{(j)} (E^{(j)})^\top. \tag{16}$$

Then our objective is to learn the multiple dictionary kernels $S^{(j)}$'s such that the mixed kernel $\widetilde{K} = \sum_j \alpha_j K^{(j)}$ will maximally correlate with the ideal kernel on the labeled samples. To achieve this we use the following objective

$$\begin{aligned} \min_{S^{(j)}\text{'s}} \; & \lambda \sum_{j=1}^{M} \left\| \alpha_j S^{(j)} - \alpha_j S_0^{(j)} \right\|_F^2 + \|\tilde{K}_l - K_l^*\|_F^2 \\ & \tilde{K}_l = \sum_j \alpha_j E_l^{(j)} S^{(j)} (E_l^{(j)})^\top \\ & s.t.\ S^{(j)} \succeq 0,\ \alpha_j \geq 0. \end{aligned} \tag{17}$$

Here $E_l^{(j)}$ are the rows in $E^{(j)}$ corresponding to labeled samples, and $\widetilde{K}_l$ are sub-block of $\widetilde{K}$ corresponding to labeled samples. The weighting coefficients α_j's are all non-negative, such that the mixed kernel $\widetilde{K}$ and its sub-block $\widetilde{K}_l$ are both PSD matrices. In the optimization problem (17), the first term states that for each kernel parameter (or a source domain), the dictionary $S^{(j)}$ should be close to the prior $S_0^{(j)}$; the second term states that the mixed kernel should be aligned with the target variables on the labeled samples.

4.1. Alternating optimization

Fixing α_j's and solving $S^{(j)}$'s. When α_j's are fixed, we will solve $S^{(j)}$'s for $j = 1, 2, ..., M$ in a cyclic manner. For each j, we have the following

$$\begin{aligned} &\min_{S^{(j)}} \lambda \left\| \alpha_j S^{(j)} - \alpha_j S_0^{(j)} \right\|_F^2 + \left\| \alpha_j E_l^{(j)} S^{(j)} (E_l^{(j)})^\top - K_{\neg j}^* \right\|_F^2 + const \\ &\quad K_{\neg j}^* = \left(K_l^* - \sum_{k \neq j} \alpha_k E_l^{(k)} S^{(k)} (E_l^{(k)})^\top \right) \\ &\quad s.t.\ \ S^{(j)} \succeq 0. \end{aligned} \tag{18}$$

By computing $\alpha_j S^{(j)}$ together as a single variable $T^{(j)}$, the resultant problem will be exactly the same as (7), namely

$$\begin{aligned} &\min_{T^{(j)}} \lambda \left\| T^{(j)} - \alpha_j S_0^{(j)} \right\|_F^2 + \left\| E_l^{(j)} T^{(j)} (E_l^{(j)})^\top - K_{\neg j}^* \right\|_F^2 \\ &\quad K_{\neg j}^* = \left(K_l^* - \sum_{k \neq j} E_l^{(k)} T^{(k)} (E_l^{(k)})^\top \right) \\ &\quad s.t.\ \ T^{(j)} \succeq 0. \end{aligned}$$

Here $S_0^{(j)}$'s are computed in a similar way as (5). Finally, the mixed kernel is simply the summation of all $E^{(j)} T^{(j)} (E^{(j)})^\top$'s.

Fixing $T^{(j)}$'s and solving α_j's. When $T^{(j)}$'s are fixed, solving α_j becomes the following

$$\begin{aligned} &\min_{\alpha_j} \left\| T^{(j)} - \alpha_j S_0^{(j)} \right\|_F^2 \\ &\quad s.t.\ \ \alpha_j \geq 0. \end{aligned}$$

The closed-form solution for α_j is

$$\alpha_j = \max \left(\frac{tr\left(S_0^{(j)} (T^{(j)})^\top \right)}{tr\left(S_0^{(j)} (S_0^{(j)})^\top \right)},\ 0 \right). \tag{19}$$

4.2. Empirical setup for multiple kernel learning

In practice, we choose $M = 5 - 10$ different kernel matrices as bases. Here M either corresponds to the number of different kernel parameters or the number of source domains. The number of landmark points is chosen as $\bar{m}$. Usually we choose $\bar{m} = m \cdot \tau / M$, where m is the number of landmark points typically chosen for single kernel case. For example, if $\tau = 3$, that means the complexity of the multiple kernel scenario would be three times as much as the single kernel case. The final learned kernel $\widetilde{K} = \sum_j E^{(j)} T^{(j)} (E^{(j)})^\top$ can be written as

$$\widetilde{K} = \mathbf{E} \cdot \mathbf{T} \cdot \mathbf{E}^\top, \tag{20}$$

where

$$\mathbf{E} = \left[E^{(1)}, E^{(2)}, ..., E^{(M)} \right], \tag{21}$$

$$\mathbf{T} = diag\left(T^{(1)}, T^{(2)}, ..., T^{(M)} \right), \tag{22}$$

which can be plugged in any kernel machine for training and testing.

For computational efficiency, we first express $\widetilde{K}$ in decomposition form, such that only a linear algorithm needs to be implemented. The details are as follows. Compute the eigenvalue decomposition $\mathbf{T} = \mathbf{U} \mathbf{\Lambda} \mathbf{U}^\top$, and let $\mathbf{Q} = \mathbf{E} \mathbf{U} \mathbf{\Lambda}^{1/2}$. As shown in [34], the final kernel SVM using kernel matrix $\tilde{K}$ can be equivalently transformed to a linear SVM using virtual samples $\mathbf{Q} = \mathbf{E} \mathbf{U} \mathbf{\Lambda}^{1/2}$. This decomposition itself is efficient since the matrix $\mathbf{T}$ is block-diagonal and therefore its eigenvalue decomposition can be performed independently inside each block, and the overall complexity is $O(M(\bar{m}^3 + n\bar{m}^2))$.

5. Related work

5.1. Low-rank decomposition

Low-rank decomposition is a powerful tool in scaling up machine learning algorithms. For example, Fine and Scheinberg [3] used low-rank approximation to solve the linear systems in interior-point iterations, which reduces the complexity of SVM from $O(n^3)$ to $O(nr^2)$. Similarly, in kernel k-means clustering [4], the per-iteration cost can be reduced from $O(n^2)$ to $O(nkr)$. Other examples include spectral clustering [6], SVM [7] and manifold learning [27]. The Nyström method provides efficient approximation to low-rank decomposition of PSD matrices, with only linear space and time complexities [5]. It reconstructs the input matrix by sampling a subset of representative rows or columns. In the literature, various sampling strategies were studied including random sampling [5,6,35], probabilistic sampling [30], k-means based sampling [7], and adaptive sampling [31,28].

5.2. Kernel learning

In the literature, many algorithms have been proposed in kernel learning [10,36,12,9,13–15]. For example, many multiple kernel learning algorithms have been proposed to search for a combination of base kernel matrices that maximizes the alignment with the "ideal kernel" $K^*(\mathbf{x}, \mathbf{z}) = y(\mathbf{x})y(\mathbf{z})$, where $y(\mathbf{x})$ is the class label of $\mathbf{x}$ [11]. However, most existing multiple kernel learning algorithms need to manipulate multiple $n \times n$ base kernel matrices, which will take at least $O(n^2)$ space and time. In comparison, the multiple-kernel-learning version of the proposed method still has a linear time and space complexity.

5.3. Low-rank decomposition and kernel learning

Although both low-rank decomposition and kernel learning have been studied extensively, works on combining matrix low-rank decomposition with kernel learning were relatively limited.

5.3.1. Cholesky with Side Information

A pioneering work that incorporates class label information in the low-rank decomposition is the Cholesky with Side Information (CSI) [2]. It iteratively selects columns of the kernel matrix that maximally reduces the hybrid of the matrix approximation error and a linear prediction error, which can significantly reduce the rank of factorization needed in a kernel classifier. The complexity is linear in the number of data points.

Our approach was motivated similarly but has important differences. First, the CSI method assumes that labels of all training instances are given (extension to semi-supervised setting is still an open problem); in comparison, we consider the more generalized semi-supervised learning scenario. Second, the CSI procedure is transductive and it does not provide a principled way to compute factorizations for new unseen samples; whereas our approach generalizes easily to new unseen samples by design.

5.3.2. Bregman kernel learning

Although kernel learning has drawn considerable interest, there are few works that provide computationally efficient learning of low rank kernel matrices. Recently, Kulis et al. [37] proposed to learn a low-rank kernel by minimizing its divergence with an initial kernel subject to distance constraints. It can automatically enforce the positive definiteness of the kernel. In particular, by using the LogDet divergence one can evaluate the kernel function over new data points. Computationally, the algorithm has a time and space complexity that is quadratic with the sample size. An earlier work [38] adopted the von-Neumann matrix divergence to learn a (full rank) kernel matrix, whose complexity is cubic with sample size.

5.3.3. Other methods

Recently, the low-rank constraint has been shown to be a useful tool for model regularization. For example, Shalit et al. [39] proposed an online learning procedure that consists of iterative gradient projections and efficient second-order retractions back to the low-rank manifold of PSD matrices. In [40], a low-rank kernel learning approach was proposed for regression by using conical combinations of base kernels and a stochastic optimization framework. The low-rank constraint was also used in matrix completion [41].

Cesa-Bianchi et al. [42] pointed out some possible limitations of approximating the kernel matrix: it may be difficult to obtain a non-trivial classification accuracy if only a limited number of kernel evaluations is possible. They found that the performance of the approximation also depends on other factors such as the loss function and the regularization parameter, which provides a new perspective in scaling up kernel methods.

5.4. Sparse Gaussian processes

Gaussian Processes (GP) are a popular tool for non-parametric Bayesian inference for nonlinear regression and classification, which can be deemed as a Bayesian version of kernel methods. Recently, sparse Gaussian process has drawn considerable interest in scalable Gaussian process inference.

Sparse Gaussian process is aimed at computing a prediction model that is composed of a small number of basis ("pseudo"-samples), in comparison with a dense model that potentially spans on all the available samples. In the regression setting, Matthias Seeger et al. [22] proposed a sparse greedy approximation for Gaussian process regression using a novel fast forward selection strategy to select candidate points, which can be interpreted as a direct approximation to the full GP marginal likelihood. Lawrence et al. [23] proposed the "Informative Vector Machine" (IVM), which used a fully randomized greedy scheme to select candidate points to include in the model, which leads to an approximation to a non-sparse GP model and competitive performance against SVM. Quiñonero-Candela and Rasmussen [24] proposed a unifying view of many existing sparse approximate Gaussian process methods by analyzing the so-called "effective prior" adopted in different methods, giving more insights on the impact of approximations.

Besides greedy selection, the locations of the pseudo-inputs can also be optimized together with other variational parameters or hyper-parameters (e.g. kernel function, noise parameter) in a principled way in learning the model. For example, Snelson and Ghahramani [19] proposed a smooth joint optimization framework to find the locations of a small, active set of point locations (also called "pseudo-inputs") and kernel/noise parameters, which can significantly reduce the computational cost of Gaussian process regression to $O(m^2 n)$ where m is the number of pseudo-inputs and n is the sample size. Titsias [20] proposed a variational formulation for sparse approximations that jointly infers the inducing inputs (as variational parameters) and the kernel hyper-parameters by maximizing a lower bound of the true log marginal likelihood. Very recently, Hensman et al. [21] proposed novel variational bounds on the marginal likelihood that allow for more efficient inference not only in regression, but also in the more challenging classification setting. In particular, one of their marginal bounds avoids introducing additional latent variables or factorizing assumptions, which combines the scalability of the stochastic optimization with the ability to optimize the positions of the inducing points. Their method has been shown to work efficiently on millions of data points. Sheth et al. [43] generalized this for general likelihood, including classification, regression, ordinal prediction, count prediction etc.

For semi-supervised learning, Lawrence and Jordan [44] proposed the Null Category Noise Model (NCNM) for semi-supervised classification, by extending the IVM method in [23] that finds the decision boundary which avoids passing through densely populated regions. Srijith et al. [45] proposed a semi-supervised ordinal regression using Gaussian Processes, by matching ordinal label distributions approximately between labeled and unlabeled data.

The important advantage of sparse Gaussian process regression is that a principled selection of the active sets (i.e., the locations of the basis vectors) can be achieved, often in conjunction with the kernel matrix (and other hyper-parameters) in a joint optimization framework thanks to the flexible Bayesian inference procedures. In other words, all the parameters can be learned simultaneously using available training samples and their labels. This can be attractive in terms of both theoretical analysis and convenience for practitioners. In comparison, our approach adopts a more heuristic, two-step procedure. In the first (unsupervised) step, representative samples are selected mainly based on the data distribution so as to better approximate the kernel matrix in the form of low-rank decomposition; in the second (semi-supervised) step, the kernel matrix defined on the representative samples is then computed using available training labels (which will be further extrapolated to the whole data set), and so a kernel-based classifier can be finally obtained using the learned kernel matrix. In other words, the training samples and their labels are utilized separately in two different but consecutive steps in our approach. Our approach has the advantage that the semi-supervised formulation is naturally easier than the one in Gaussian Process. On the other hand, the GP approach works with general likelihoods, not just classification, whereas our use of the kernel alignment is limited to classification.

6. Experiments

In this section, we have performed extensive experiments to compare the proposed methods with state-of-the-arts techniques across different benchmark data sets. We report our experimental results for single kernel setting and multiple kernel setting in Section 6.1 and Section 6.2, respectively.

6.1. Single kernel setting

This section compares 8 algorithms on learning low-rank kernel: (1) LibSVM [46]: support vector machine, which uses only the small number of labeled samples for training; (2) Nyström: standard Nyström method; (3) CSI: Cholesky with Side Information [2]; (4) Cluster: cluster kernel [47]; (5) Spectral: non-parametric spectral graph kernel [17]; (6) Breg: low-rank kernel learning with Bregman divergence [37]; (7) NCNM [44]: Sparse Gaussian Process for Semi-supervised Learning; and (8) Our proposed method. Most algorithms can learn the $n \times n$ low-rank kernel[2] matrix on labeled and unlabeled samples[3] in the form of $K = GG^\top$, which can then be fed into an SVM for classification. The resultant problem will be a linear SVM using G as training/testing samples [34].

[2] The rank of the learned kernel matrix is chosen to be 10% of sample size (or a fixed number if sample is too large).

[3] Method (3) uses some heuristics to compute the kernel matrix between labeled and unlabeled samples, since only labeled samples are used in training.

Table 1
Performance of single kernel learning methods; top row: mean/std of error (%); bottom row: average time (in seconds).

Data $n/d/\#cls$	LibSVM	Nyström	CSI	Spectral	Cluster	Breg	NCNM	Ours single kernel
g241C	22.09±1.34	27.09±2.16	22.37±1.80	23.64±1.28	26.59±3.96	26.31±1.81	22.86±1.51	**21.57±0.85**
1500/241/2	0.9	0.8	0.8	108.2	63.3	17.7	22.7	1.2
Digit1	4.96±1.15	5.67±1.25	4.97±0.84	**3.87±1.71**	5.53±1.01	6.01±1.62	6.40±1.03	4.71±0.71
1500/241/2	0.1	0.8	0.8	70.5	0.5	27.3	22.3	1.2
USPS	11.55 ± 4.33	10.67±3.46	8.60±2.55	11.28±0.51	**6.96±1.14**	11.45±2.34	11.29±3.49	8.66±1.14
1500/241/2	0.1	0.7	0.7	78.4	149.1	27.5	22.4	1.0
coil	**11.00±1.14**	19.04±2.90	19.46±2.88	31.21±10.06	19.38±2.40	19.57±3.41	19.19±7.37	19.60±3.20
1500/241/6	0.2	0.8	0.8	31.0	41.7	19.8	198.4	1.2
coil2	19.82±4.10	12.98±4.60	12.44±4.68	14.57±3.28	13.61±2.24	14.58±3.17	12.99±3.51	**11.83±3.40**
1500/241/2	0.1	0.8	0.6	39.1	47.6	27.3	22.3	1.1
Text	26.43±3.00	27.10±2.22	22.69±1.86	24.90±2.04	27.8±2.19	23.66±0.90	22.71±1.10	**22.10±1.32**
1500/11960/2	2.82	37.9	4.9	40.5	287.9	44.3	780	28.2
german	35.18±1.35	40.31±2.74	37.73±2.27	33.83±10.51	**31.90±3.42**	38.42±2.95	35.59±2.51	36.84±3.37
1000/24/2	0.1	0.2	0.6	41.6	2.5	512.3	5.19	0.4
usps49	3.21±1.20	2.73±0.72	2.36±1.04	**1.58±0.40**	1.74±0.23	3.04±0.39	1.83±0.51	**1.67±0.34**
1296/256/2	0.1	0.7	0.8	26.6	8.4	18.5	17.63	1.0
usps27	1.41±0.22	1.25±0.27	1.29±0.27	1.98±0.33	1.25±0.31	1.52±0.47	1.21±0.24	**1.10±0.30**
1367/256/2	0.1	0.6	0.6	7.82	14.6	22.5	19.9	1.0
adult1a	26.22±1.53	29.33±2.96	25.69±2.30	27.27±1.99	24.50±2.68	32.62±2.22	24.82±2.67	**23.93±2.06**
1605/123/2	0.1	0.6	0.7	32.4	14.1	17.5	17.6	1.2
dna	**13.13±0.63**	15.92±2.03	15.45±1.36	15.68±1.51	20.87±2.16	15.80±0.16	14.21±1.06	15.50±1.83
2000/180/3	0.2	1.8	1.2	62.2	48.5	38.1	108.2	2.4
segment	8.23±0.76	9.60±1.49	9.39±1.14	15.51±3.10	17.91±2.82	10.08±1.79	**5.45±0.49**	9.59±1.20
2310/29/7	0.1	1.7	2.1	83.2	19.9	22.6	141.1	2.7
svmgd1a	6.59±1.18	5.22±0.97	6.55±0.33	6.54±0.91	5.18±1.79	5.51±1.45	10.68±8.57	**4.40±0.83**
3089/4/2	0.1	0.6	4.5	205.5	47.6	17.6	19.9	1.5
satimage	15.11±0.66	18.70±1.82	18.54±0.97	19.39±1.63	20.78±2.36	18.52±1.82	**14.08±1.02**	17.88±1.40
6435/36/6	0.1	1.7	2.2	285.5	197.7	638.4	238.97	2.8
usps-full	**6.01±0.45**	14.47±1.43	**13.43±1.51**	14.32±1.81	14.79±2.39	14.25±2.29	–	13.68±1.42
7291/256/10	2.7	5.5	3.8	521.3	363.9	1418.5	–	6.1
mnist	34.68±3.74	25.12±1.85	24.70±1.26	–	–	23.96±1.73	–	**21.85±1.77**
70000/780/10	8.1	80.4	33.3	–	–	151.7	–	82.3

We use benchmark data sets from the SSL data sets [48][4] and the LIBSVM data.[5] For each data set, we randomly pick 100 labeled samples evenly among all classes and use the remaining samples as unlabeled data, repeat 30 times, and report the averaged classification error on unlabeled data. We use the Gaussian kernel $K(\mathbf{x}_1, \mathbf{x}_2) = \exp(-\|\mathbf{x}_1 - \mathbf{x}_2\|^2/b)$. Parameter selection is difficult in semi-supervised learning, so, we chose the kernel width as the averaged pairwise squared distances between samples because it empirically gives reasonable performance. For the regularization parameter C in linear SVM, we use the heuristic implemented in LIBLINEAR package [49]. Most implementations are in Matlab (for method (3) we use implementation in [2] with core functions written in C) and run on a PC with 2G memory and 2.8 GHz processor.

Results are reported in Table 1. Methods statistically better than others with a confidence level that is at least 95% (paired t-test) are highlighted. Method (1) and (2) are baseline methods that do not involve any kernel learning, therefore they are computationally very efficient. Note that Method (1) only uses labeled samples for training, so it has higher error rates. Method (3) is computationally efficient as well because it only uses labeled samples for training. Method (4) and (5) require eigenvalue decomposition of the kernel matrix (or graph Laplacian), therefore they are computationally more expensive. Method (6) and (7) are observed to require a relatively larger number of iterations to converge. Our approach is very efficient and can be orders of magnitudes faster than methods (4), (5), (6) and (7).

From Table 1, we can see that on most data sets, algorithms using labels in kernel learning outperform the baseline algorithm (standard Nyström), indicating the value of side information. Our approach is competitive with state-of-the-art kernel learning algorithms. On the largest data set mnist, method (4), (5), (7) can not run on our PC due to huge memory consumption; in comparison, our approach is very efficient and gives the lowest error rate on this data set. We also examine the alignment score (15) used to choose the hyper-parameter λ in Fig. 1. As can be seen, the alignment score correlates nicely with the classification accuracy on unlabeled data.

6.2. Multiple kernels setting

In this section we explore the multiple kernel learning scenario. The data sets used for evaluating the performance are summarized in Table 2. For each data set, we randomly pick 100 labeled samples evenly among all classes, repeat

[4] The data sets are from http://olivier.chapelle.cc/ssl-book/benchmarks.html.
[5] The data sets are from http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/.

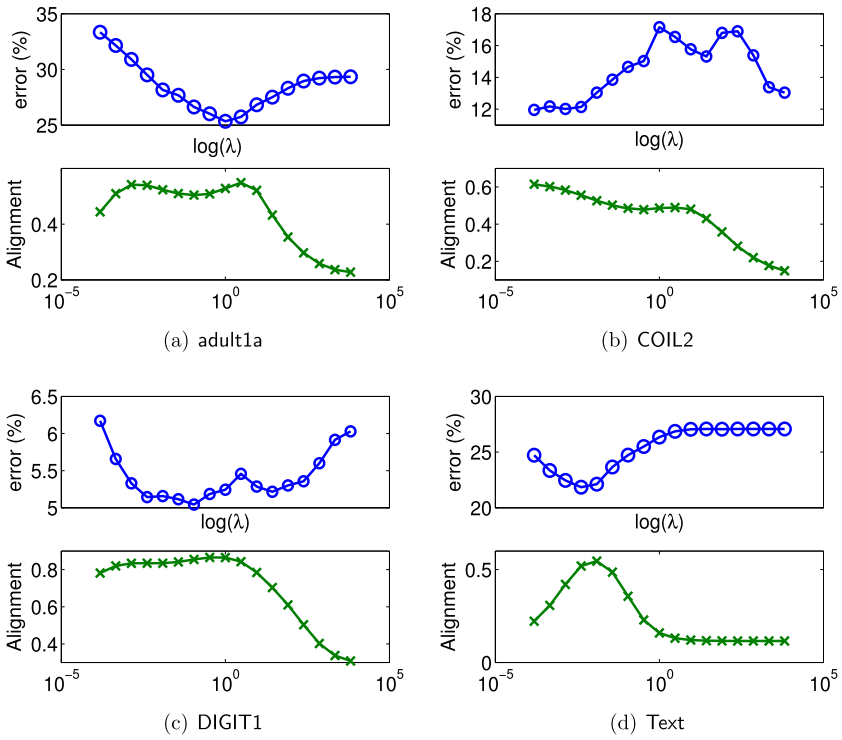


Fig. 1. The alignment score defined in (15) (bottom box) correlates with the classification error on unlabeled data (top box). By choosing the λ that gives the highest alignment, we can obtain a satisfactory prediction.

30 times, and report the averaged classification error on unlabeled data. By changing the RBF kernel parameter b used in the Gaussian kernel $K(\mathbf{x}_1, \mathbf{x}_2) = \exp(-\|\mathbf{x}_1 - \mathbf{x}_2\|^2/b)$ from $\{2^{-4}b_0, 2^{-3}b_0, \ldots, 2^3 b_0, 2^4 b_0\}$, where b_0 is the average pairwise squared distances between samples, we choose altogether 9 base kernel matrices.

We compare our proposed method with the following 5 state-of-the-art multiple kernel learning algorithms[6]: (1) ABMKL: Alignment based Multiple Kernel Learning [9]; (2) CABMKL: Center Alignment based Multiple Kernel Learning [16]; (3) Support Kernel Machines (SKM): a SMO-like algorithm for multiple kernel learning proposed by Bach et al. [13]; (4) SimpleMKL: an alternating optimization approach for multiple kernel learning [14] and (5) GLMKL: Group lasso-based Multiple Kernel Learning [15]. We set the regularization parameter C in SVM to 1 for all algorithms. The parameter λ for our proposed method is chosen from $\{10^{-5}, 10^{-4}, \ldots, 10^4, 10^5\}$. Similarly to (15), the optimal λ for multiple kernel learning is also selected based on the alignment score as follows,

$$\lambda^* = \arg\max_{\lambda \in \mathcal{G}} \prod_{j=1}^{M} \rho\left[\alpha_j S^{(j)}(\lambda), \alpha_j S_0^{(j)}\right] \cdot \rho\left[\sum_{i=1}^{M} \alpha_i E_l{}^{(i)} S^{(i)}(\lambda) E_l{}^{(i)\top}, K_l^*\right].$$

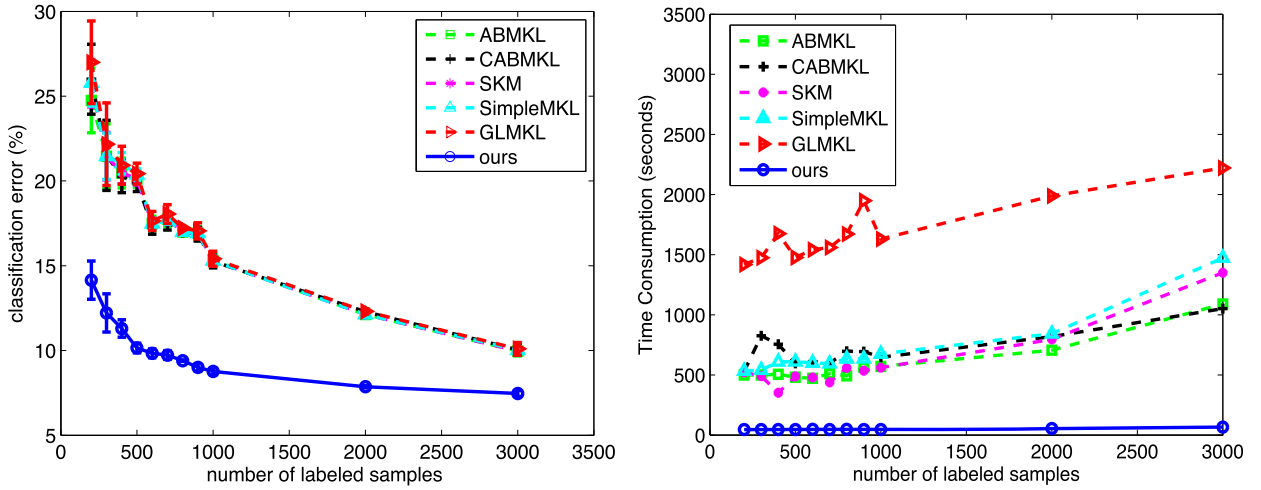
Experimental results are reported in Table 2. The computing time is evaluated on a laptop with 4 G memory and Intel® Core™ i5-3317U CPU@1.70 GHz. For each data set, the methods statistically better than others with a confidence level that is at 95% are highlighted in bold. As shown in Table 2, our proposed method achieves the lowest classification error on 11 out of 17 data sets. For the other 6 data sets, our method is the second best in 5 data sets (i.e., USPS, german, usps49, usps27 and svmgd1a). Overall, our method performs better than other multiple kernel learning algorithms. In terms of computing time, our method is much faster than other methods on large datasets with sample size larger than 6,000 (i.e., satimage, usps-full, mnist, webspam).

The effect of number of labeled samples. We use the mnist dataset with 70,000 samples to study how the computing time and classification error rate changes when the number of labeled samples increases. In our experimental setting, the number of labeled samples varies from $l = \{200, 300, 400, \ldots, 1000, 2000, 3000\}$. We use the remaining $n - l$ ($n = 70{,}000$) unlabeled samples to evaluate the classification error rate. As shown in the subfigure (a) of Fig. 2, the error rate decreases when the number of labeled samples increases. The error rates obtained by our methods on mnist dataset are significantly lower than other multiple kernel learning algorithms. The subfigure (b) of Fig. 2 clearly shows our method are several times faster than other algorithms. In our experimental setting, the computing time is dominated by prediction time because

[6] The implementation of these 5 multiple kernel learning algorithms can be downloaded from http://users.ics.aalto.fi/gonen/jmlr11.php.

Table 2
Performance of multiple kernel learning methods;top row: mean/std of error (%); bottom row: average time (in seconds).

Data size/dim/#cls	ABMKL	CABMKL	SKM	SimpleMKL	GLMKL	Ours multiple kernels
g241C	27.19±8.76	23.47±1.09	27.49±6.79	27.55±6.74	21.90±1.37	**18.08±1.12**
1500/241/2	1.1	0.4	0.7	0.7	0.7	0.6
Digit1	6.17±0.82	6.86±1.25	6.49±0.91	6.49±0.91	6.39±0.90	**4.77±0.93**
1500/241/2	0.8	0.2	0.2	0.6	0.9	0.7
USPS	17.59±4.02	**8.64±2.64**	13.04±4.27	13.01±4.23	14.03±4.55	10.14±3.01
1500/241/2	0.4	0.2	0.3	0.7	0.8	0.7
coil	15.71±1.55	22.04±3.94	14.09±1.46	12.15±1.51	14.30±1.44	**10.38±0.97**
1500/241/6	0.9	0.1	3.4	4.1	6.1	0.8
coil2	18.11±2.23	**11.76±2.65**	19.04±2.61	18.61±3.23	18.81±4.52	**12.16±2.77**
1500/241/2	0.4	0.4	0.4	0.6	0.8	0.6
Text	27.62±1.41	23.29±1.61	27.50±1.52	28.14±2.19	27.50±1.52	**22.54±1.49**
1500/11960/2	6.1	5.2	5.2	5.5	20.4	11.1
german	35.80±2.64	**33.93±3.20**	36.41±2.79	36.29±2.78	36.47±2.85	**34.83±2.84**
1000/24/2	0.1	0.1	0.1	0.2	0.2	0.2
usps49	9.78±1.66	**1.82±0.73**	5.39±1.39	5.59±1.3	5.7±1.5	2.1±0.6
1296/256/2	0.3	0.2	0.2	0.7	0.7	0.5
usps27	3.66±1.41	**1.24±0.21**	3.50±1.41	3.50±1.41	3.51±1.41	1.42±0.33
1367/256/2	0.3	0.2	0.2	0.4	0.7	0.5
adult1a	30.90±9.81	26.62±3.57	25.98±1.98	**25.88±2.42**	26.38±1.63	**25.41±2.55**
1605/123/2	0.3	0.2	0.3	0.6	0.7	0.6
dna	13.30±0.70	18.15±1.94	14.27±1.10	14.29±1.10	14.59±1.51	**12.89±0.55**
2000/180/3	1.7	0.1	1.3	1.3	3.6	1.0
segment	9.24±0.74	9.50±1.32	9.27±1.04	9.18±0.98	9.26±0.97	**6.56±0.92**
2310/29/7	4.0	3.2	5.8	3.4	6.1	1.2
svmgd1a	7.02±1.55	**4.80±0.84**	6.79±1.21	6.67±1.21	6.74±1.26	**5.05±1.52**
3089/4/2	0.2	0.5	0.4	0.3	0.8	0.9
satimage	**13.60±0.71**	17.13±0.86	14.03±0.65	14.04±0.67	14.29±0.59	14.93±1.01
6435/36/6	7.8	5.9	8.6	3.6	12.9	1.5
usps-full	7.73±0.62	13.93±1.59	7.83±0.70	7.92±0.81	8.10±0.69	**6.37±0.47**
7291/256/10	15.4	21.0	23.2	20.9	52.1	3.2
webspam	24.83±2.87	16.20±0.99	23.54±2.51	16.86±0.83	22.94±2.59	**15.67±1.08**
350000/254/2	125.1	165.6	153.9	164.3	319.9	83.2
mnist	29.74±2.30	28.48±2.15	26.47±2.41	29.38±1.87	28.58±1.87	**20.74±2.08**
70000/780/10	529.9	868.6	543.4	663.7	1574.4	45.1



(a) Error Rate (b) Time Consumption

Fig. 2. Evaluation of multiple kernel learning algorithms using different number of labeled samples.

the number of labeled samples is very small compared to the data size. So, as shown in the subfigure (b) of Fig. 2, the computing time of other multiple kernel learning algorithms roughly increases linearly with the number of labeled samples. The computing time of our proposed method do not increase because we fixed the number of landmark points (i.e., the rank of the learned kernel) at 100.

Table A1
Performance of two methods for choosing scaling factor β; top row: mean/std of error (%); bottom row: average time (in seconds).

Data size/dim/#cls	Heuristic	Iterative optimization	Data	Heuristic	Iterative optimization
g241C	**21.57±0.85**	21.76±0.54	usps27	**1.10±0.30**	**1.05±0.34**
1500/241/2	1.2	1.5	1367/256/2	1.0	1.3
Digit1	4.71±0.71	**4.64±0.48**	adult1a	23.93±2.06	**22.74±2.12**
1500/241/2	1.2	1.5	1605/123/2	1.2	5.1
USPS	8.66±1.14	**8.53±1.27**	dna	**15.50±1.83**	15.87±1.94
1500/241/2	1.0	1.7	2000/180/3	2.4	3.0
coil	**19.60±3.20**	20.10±3.06	segment	**9.59±1.20**	10.02±2.13
1500/241/6	1.2	3.3	2310/29/7	2.7	3.2
coil2	11.83±3.40	**11.75±3.71**	svmgd1a	4.40±0.83	**4.12±0.69**
1500/241/2	1.1	2.5	3089/4/2	1.5	2.7
Text	22.10±1.32	**21.86±1.43**	satimage	17.88±1.40	**16.40±1.32**
1500/11960/2	28.2	40.5	6435/36/6	2.8	3.2
german	**36.84±3.37**	37.83±4.14	usps-full	**13.68±1.42**	**13.54± 1.53**
1000/24/2	0.4	1.0	7291/256/10	6.1	9.2
usps49	1.67±0.34	**1.60±0.43**	mnist	**21.85±1.77**	**21.03±1.24**
1296/256/2	1.0	1.9	70000/780/10	82.3	177.3

7. Conclusions

In this paper, we proposed an efficient kernel low-rank decomposition algorithm called the generalized Nyström method. Our approach inherits the advantages from both low-rank matrix decomposition and kernel learning. Our approach is computationally very efficient and the space and time complexity is both linear in the sample size and dimension. Our approach can also incorporate side information to compute low-rank decomposition that aligns well with the learning target, leading to improved generalization performance. Our approach is endowed with a flexible non-parametric structure and can extend naturally to new data. It shows significant performance gains in benchmark learning tasks. In the future, we will consider learning a *sparse* dictionary kernel in our method. In case a large number of landmark points are selected, sparse dictionary kernel can effectively reduce the computational cost while preserving the model capacity. We are also interested in incorporating do-not-link constraints in the loss function for improved generalization performance. Finally, we are also pursuing more principled ways of determining the rank parameter by achieving a balanced tradeoff between computational savings and the generalization performance.

Acknowledgements

The authors would like to thank the associate editor and all the reviewers for the valuable comments. Hongyuan Zha would like to acknowledge the support of research funding STCSM 15JC1401700 and NSFC IIS-1639792.

Appendix A. Comparison of different choices of β

In this section, we present our empirical results on testing two different methods for choosing the scaling factor β in (5): (1) A heuristic way as shown in (8); and (2) Iterative optimization of β with S in (7). The experimental setting is the same as in section 6.1. Our empirical results are summarized in Table A1. Overall speaking, the iterative approach can be a little more accurate but computationally more expensive.

References

[1] C. Williams, M. Seeger, The effect of the input density distribution on kernel-based classifiers, in: Proceedings of the 17th International Conference on Machine Learning, 2000, pp. 1159–1166.
[2] F.R. Bach, M.I. Jordan, Predictive low-rank decomposition for kernel methods, in: Proceedings of the 22nd International Conference on Machine Learning, 2005, pp. 33–40.
[3] S. Fine, K. Scheinberg, Efficient SVM training using low-rank kernel representations, J. Mach. Learn. Res. 2 (2001) 243–264.
[4] B. Kulis, S. Basu, I. Dhillon, R. Mooney, Semi-supervised graph clustering: a kernel approach, in: Proceedings of the 22nd International Conference on Machine Learning, 2005, pp. 457–464.
[5] C. Williams, M. Seeger, Using the Nyström method to speed up kernel machines, in: Advances in Neural Information Processing Systems, vol. 13, 2001, pp. 682–690.
[6] C. Fowlkes, S. Belongie, F. Chung, J. Malik, Spectral grouping using the Nyström method, IEEE Trans. Pattern Anal. Mach. Intell. 26 (2) (2004) 214–225.
[7] K. Zhang, I.W. Tsang, J.T. Kwok, Improved Nyström low-rank approximation and error analysis, in: Proceedings of the 25th International Conference on Machine Learning, 2008, pp. 1232–1239.
[8] F. Bach, M. Jordan, Kernel independent component analysis, J. Mach. Learn. Res. 3 (2002) 1–48.
[9] G.R. Lanckriet, N. Cristianini, P. Bartlett, L.E. Ghaoui, M.I. Jordan, Learning the kernel matrix with semidefinite programming, J. Mach. Learn. Res. 5 (2004) 27–72.
[10] N. Cristianini, J. Shawe-Taylor, A. Elisseeff, J.S. Kandola, On kernel-target alignment, in: Advances in Neural Information Processing Systems, 2002, pp. 367–373.

[11] J.T. Kwok, I.W. Tsang, Learning with idealized kernels, in: Proceedings of the 20th International Conference on Machine Learning, 2003, pp. 400–407.
[12] K. Sinha, M. Belkin, Semi-supervised learning using sparse eigenfunction bases, in: Advances in Neural Information Processing Systems, 2009, pp. 1687–1695.
[13] F.R. Bach, G.R. Lanckriet, M.I. Jordan, Multiple kernel learning, conic duality, and the SMO algorithm, in: Proceedings of the 21st International Conference on Machine Learning, vol. 6, 2004.
[14] A. Rakotomamonjy, F. Bach, S. Canu, Y. Grandvalet, SimpleMKL, J. Mach. Learn. Res. 9 (2008) 2491–2521.
[15] Z. Xu, R. Jin, H. Yang, I. King, M.R. Lyu, Simple and efficient multiple kernel learning by group lasso, in: Proceedings of the 27th International Conference on Machine Learning, 2010, pp. 1175–1182.
[16] C. Cortes, M. Mohri, A. Rostamizadeh, Two-stage learning kernel algorithms, in: Proceedings of the 27th International Conference on Machine Learning, 2010, pp. 239–246.
[17] X. Zhu, J. Kandola, Z. Ghahramani, J.D. Lafferty, Nonparametric transforms of graph kernels for semi-supervised learning, in: Advances in Neural Information Processing Systems, 2004, pp. 1641–1648.
[18] K. Zhang, J.T. Kwok, Clustered Nyström method for large scale manifold learning and dimension reduction, IEEE Trans. Neural Netw. 21 (2010) 1576–1587.
[19] E. Snelson, Z. Ghahramani, Sparse Gaussian processes using pseudo-inputs, in: Advances in Neural Information Processing Systems, 2005, pp. 1257–1264.
[20] M.K. Titsias, Variational learning of inducing variables in sparse Gaussian processes, in: International Conference on Artificial Intelligence and Statistics, vol. 12, 2009, pp. 567–574.
[21] J. Hensman, A. Matthews, Z. Ghahramani, Scalable variational Gaussian process classification, in: Proceedings of the 18th International Conference on Artificial Intelligence and Statistics, 2015, pp. 351–360.
[22] M. Matthias Seeger, C. Williams, N. Lawrence, Fast forward selection to speed up sparse Gaussian process regression, in: Artificial Intelligence and Statistics, 2003.
[23] N.D. Lawrence, M. Seeger, R. Herbrich, Fast sparse Gaussian process methods: the informative vector machine, in: Advances in Neural Information Processing Systems, 2003.
[24] J. Quiñonero-Candela, C.E. Rasmussen, A unifying view of sparse approximate Gaussian process regression, J. Mach. Learn. Res. 6 (Dec. 2005) 1939–1959.
[25] K. Zhang, L. Lan, J. Liu, A. Rauber, F. Moerchen, Inductive kernel low-rank decomposition with priors: a generalized Nyström method, in: Proceedings of the 29th International Conference on Machine Learning, 2012, pp. 305–312.
[26] W. Hackbusch, Integral Equations: Theory and Numerical Treatment, vol. 120, Springer Science & Business Media, 1995.
[27] A. Talwalkar, S. Kumar, H. Rowley, Large-scale manifold learning, in: IEEE Conference on Computer Vision and Pattern Recognition, 2008, pp. 1–8.
[28] A.J. Smola, B. Schökopf, Sparse greedy matrix approximation for machine learning, in: Proceedings of the 17th International Conference on Machine Learning, 2000, pp. 911–918.
[29] S. Kumar, M. Mohri, A. Talwalkar, On sampling-based approximate spectral decomposition, in: Proceedings of the 26th International Conference on Machine Learning, 2009, pp. 553–560.
[30] P. Drineas, M. Mahoney, On the Nystrom method for approximating a gram matrix for improved kernel-based learning, J. Mach. Learn. Res. 6 (2005) 2153–2175.
[31] S. Kumar, M. Mohri, A. Talwalkar, Sampling methods for the Nyström method, J. Mach. Learn. Res. 13 (1) (2012) 981–1006.
[32] K. Kiri Wagstaff, C. Cardie, S. Rogers, S. Schroedl, Constrained K-means clustering with background knowledge, in: Proceedings of the Eighteenth International Conference on Machine Learning, 2001, pp. 577–584.
[33] A. Nemirovski, Efficient Methods in Convex Programming, Lecture Notes, 1994.
[34] K. Zhang, L. Lan, Z. Wang, F. Moerchen, Scaling up kernel SVM on limited resources: a low-rank linearization approach, in: International Conference on Artificial Intelligence and Statistics, 2012, pp. 1425–1434.
[35] D. Achlioptas, F. McSherry, Fast computation of low rank matrix approximations, in: Proceedings of the 23rd ACM Symposium on Theory of Computing, 2001, pp. 611–618.
[36] M. Gönen, E. Alpaydin, Multiple kernel learning algorithms, J. Mach. Learn. Res. 12 (2011) 2211–2268.
[37] B. Kulis, M. Sustik, I. Dhillon, Low-rank kernel learning with Bregman matrix divergences, J. Mach. Learn. Res. 10 (2009) 341–376.
[38] K. Tsuda, G. Rätsch, M.K. Warmuth, Matrix exponentiated gradient updates for on-line learning and Bregman projection, J. Mach. Learn. Res. (2005) 995–1018.
[39] U. Shalit, D. Weinshall, G. Chechik, Online learning in the manifold of low-rank matrices, in: Advances in Neural Information Processing Systems, 2010, pp. 2128–2136.
[40] P. Machart, T. Peel, S. Anthoine, L. Ralaivola, H. Glotin, Stochastic low-rank kernel learning for regression, in: Proceedings of the 28th International Conference on Machine Learning, 2011, pp. 969–976.
[41] E. Candés, B. Recht, Exact matrix completion via convex optimization, Found. Comput. Math. 9 (6) (2009) 717–772.
[42] N. Cesa-Bianchi, Y. Mansour, O. Shamir, On the complexity of learning with kernels, in: Proceedings of the 28th Conference on Learning Theory, 2015, pp. 297–325.
[43] R. Sheth, Y. Wang, R. Khardon, Sparse variational inference for generalized Gaussian process models, in: Proceedings of the 32nd International Conference on Machine Learning, 2015, pp. 1302–1311.
[44] N.D. Lawrence, M.I. Jordan, Semi-supervised learning via Gaussian processes, in: Advances in Neural Information Processing Systems, 2004, pp. 753–760.
[45] P. Srijith, S. Shevade, S. Sundararajan, Semi-supervised Gaussian process ordinal regression, in: Proceedings of European Conference of Machine Learning, 2013.
[46] C.-C. Chang, C.-J. Lin, LIBSVM: a library for support vector machines, ACM Trans. Intell. Syst. Technol. 2 (3) (2011) 27.
[47] O. Chapelle, J. Weston, B. Schölkopf, Cluster kernels for semi-supervised learning, in: Advances in Neural Information Processing Systems, 2002, pp. 585–592.
[48] O. Chapelle, B. Schölkopf, A. Zien, Semi-Supervised Learning, MIT Press, Cambridge, 2006.
[49] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, C.-J. Lin, LIBLINEAR: a library for large linear classification, J. Mach. Learn. Res. 9 (2008) 1871–1874.