Information Sciences 370–371 (2016) 50–62

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

Efficient learning of supervised kernels with a graph-based loss function

Binbin Pan[a,b], Wen-Sheng Chen[a,b,*], Bo Chen[a,b], Chen Xu[c]

[a] College of Mathematics and Statistics, Shenzhen University, Shenzhen 518060, China
[b] Shenzhen Key Laboratory of Media Security, Shenzhen University, Shenzhen 518060, China
[c] Institute of Intelligent Computing Science, Shenzhen University, Shenzhen 518060, China

ARTICLE INFO

Article history:
Received 3 September 2015
Revised 11 July 2016
Accepted 25 July 2016
Available online 27 July 2016

Keywords:
Kernel learning
Kernel methods
Side information
Loss function
Supervised learning

ABSTRACT

This paper presents our study of the problems associated with learning supervised kernels from a large amount of side information. We propose a new loss function derived from the Laplacian matrix of a special complete graph that is generated from the side information. We analyze the relationship between the proposed loss function and the kernel alignment. Our theoretical analysis shows that the proposed loss function has a close relationship with kernel alignment, that is, they both make use of side information that is fused in a matrix, in addition to a similar regularization strategy. Moreover, the proposed loss function has a linear form, and thus it is more efficient in learning side information than kernel alignment that has to be performed nonlinearly. The proposed loss function is used to generate new kernels as "low-cost" alternatives of kernels learned by certain state-of-the-art methods. The empirical results demonstrate the superiority of the proposed method over state-of-the-art methods in terms of classification accuracy and computational cost.

1. Introduction

In recent years, kernel methods have been widely applied to pattern recognition [37], image analysis [28], computer vision [4], and so on. Kernel methods are a class of algorithms that measure the similarities among data points via a kernel function (or simply a *kernel*), which is a similarity function over pairs of data points. The most popular example of kernel methods is the Support Vector Machine (SVM) [7,29,34]. Kernels play an important role in the performance of kernel methods. In contrast to manual selection of a kernel, an automatic method, given a specific data set and a specific application domain, is more valuable in practice, since it requires less prior knowledge from users. Lately, a few studies [5,8,11,12,14,18–23,27,32,35,36,38–41] focused on automatic kernel learning based on the side information of data. The kernels that encode the side information of data are referred to as supervised kernels. A large amount of side information exists in many training data sets collected for real-life applications. In face recognition, for example, a training data set often contains hundreds of persons, each of whom has several instances captured under various imaging conditions. The number of labels could be as high as thousands. One important problem related to learning supervised kernels is to determine how to make use of the side information effectively and efficiently. In general, there are two ways to utilize the side information. The first is to

* Corresponding author.
E-mail addresses: pbb@szu.edu.cn (B. Pan), chenws@szu.edu.cn (W.-S. Chen), chenbo@szu.edu.cn (B. Chen), xuchen@szu.edu.cn (C. Xu).

http://dx.doi.org/10.1016/j.ins.2016.07.064
0020-0255/

learn the side information by maximizing the kernel alignment, i.e., the alignment of the combined kernel with the available labels [8,19,39,40]. The other is to incorporate the side information by pairwise constraints [14,18,22].

Kernel alignment [8] aims to measure the similarity between the learned kernel and the target kernel derived from the labels. Kernel alignment can be viewed as a loss function that measures the interpretation cost of a kernel with respect to a given training set. Some algorithms were presented for learning supervised kernels based on kernel alignment [19,39,40]. All these algorithms maximize kernel alignment as an objective function along with additional constraints. The optimization problems are often formulated as Quadratically Constrained Quadratic Programming (QCQP) because of the quadratic form of the kernel alignment. Interior-point algorithms can be applied to solve the QCQP problem effectively. However, they have poor scalability, i.e., the computational time these algorithms require to process a large-scale or even medium-scale data set increases dramatically. Therefore, the nonlinear form of kernel alignment often complicates the optimization problem unnecessarily although the optimization problem involves linear constraints only. Moreover, kernel alignment is limited to class labels, i.e., it is not applicable to other types of side information such as pairwise similarity/dissimilarity. In contrast to kernel alignment, pairwise constraints incorporate side-information as linear constraints, which are able to process different types of side information [14,18,22]. The learning problems are often formulated as an optimization problem where the objective function is the Bregman matrix divergence and the constraints are linear constraints induced by pairwise constraints. The Bregman matrix divergence measures the discrepancy between the learned kernel matrix and an input kernel matrix. Such problems can be solved via the Bregman projection algorithm, which randomly chooses one constraint and solves the resulting problem exactly. Convergence requires every constraint to be visited several times. When a large amount of side information is available, it is quite possible that the number of constraints is of the same magnitude, or even much larger than the number of data points. In this case, it is computationally expensive to solve these problems.

In this paper, we propose a new loss function capable of exploiting side information efficiently for supervised kernel learning. The loss function is defined based on a complete and weighted graph where the weight of each edge is contributed by side information. We named this graph, which encodes the side information, the Supervised Complete Graph (SCG). The learned kernel is required to be smooth over the SCG. This means that a pair of data points is expected to be close in the kernel space if they belong to the same class. The resulted loss function is named *scg-loss*. We highlight the contributions of this paper as follows.

- Our scg-loss has two attractive properties. It is a linear function of the kernel matrix to be learned, and can be applied to various forms of side information such as class labels and pairwise similarity/dissimilarity. The linear form of an scg-loss allows us to develop efficient algorithms. To the best of our knowledge, none of the existing loss functions have these two properties simultaneously.
- We theoretically compare scg-loss with kernel alignment. It demonstrates that both scg-loss and kernel alignment aim at maximizing the inner product between a kernel matrix and the target matrix derived from side-information under a regularization scheme that controls the trace of the kernel matrix.
- The scg-loss is used to develop two new kernels as "low-cost" alternatives of kernels output by two existing kernel learning algorithms [14,40]. One resulting optimization problem is a Linear Programming (LP) problem that can be solved efficiently. The other problem leads to a closed-form solution and the core operation involved is matrix inversion.
- The experimental results indicate that algorithms using scg-loss are significantly more efficient than competing methods, while achieving comparable accuracy. More importantly, the computational complexity is nearly unchanged when more side information is utilized. This is important for dealing with a large amount of side information.

The remainder of the paper is organized as follows. Section 2 briefly reviews some related work for learning supervised kernels. Our loss function is presented in Section 3. A theoretical analysis of the proposed loss function is provided in Section 4. In Section 5, new kernels are developed using the proposed loss function. Experimental results are reported in Section 6 and conclusions are drawn in Section 7.

2. Related work

We firstly review the literature on supervised kernel learning. Then we briefly introduce the algorithms for learning supervised kernels using kernel alignment and pairwise constraints.

2.1. Supervised kernel learning

We are given a set of data points $X = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$ and the associated side information. The side information is generally provided in two forms: class label and pairwise similarity/dissimilarity. For class label, each labeled data $\mathbf{x}_i$ is assigned to a label y_i. For pairwise similarity/dissimilarity, we collect two sets $\mathcal{S}$ and $\mathcal{D}$, where $\mathcal{S}$ contains the similar pairwise data and $\mathcal{D}$ includes the dissimilar pairwise data. Note that one can build pairwise similarity/dissimilarity from the class label, i.e., two data points are similar if they have the same label and dissimilar otherwise. Thus, pairwise similarity/dissimilarity is more general. The aim of supervised kernel learning is to learn a kernel matrix or a kernel function using the available data and side information.

Algorithms for learning supervised kernels can be roughly classified into two categories. The first category incorporates the side information by optimizing a loss function. The loss functions used in previous work are listed in Table 1. For each

Table 1
List of the loss functions. "linear" indicates whether the loss function is a linear function of the kernel matrix. "Pairwise similarity" means whether the loss function can utilize pairwise similarity/dissimilarity. The scg-loss is the loss function proposed in this paper.

Loss function	Linear	Pairwise similarity	Literatures
Kernel alignment	×	×	[19,39,40]
Fisher score	×	×	[21,23,36]
Hinge-loss	×	×	[12,19,38]
Squared-loss	×	×	[5]
Hoi-loss	×	√	[11,41]
Log-loss	×	√	[20]
Hilbert-Schmidt independent criterion	√	×	[32]
Ideal regularization	√	×	[27]
Centered kernel polarization	√	×	[35]
Scg-loss	√	√	new

loss function, we indicate whether it is a linear function of the kernel matrix, and whether it can be applied to pairwise similarity/dissimilarity.

The kernel alignment is of nonlinear form and only restricted to the class label. A similar situation exists in the Fisher score, which maximizes the ratio of between-class scatter to within-class scatter in kernel space. Hinge-loss and Squared-loss also have nonlinear forms and can only be applied to the class label. Hoi-loss and log-loss can utilize various forms of side information, but they are both nonlinear and thus often complicate the optimization problems. The Hilbert-Schmidt independent criterion, ideal regularization, and centered kernel polarization are linear forms that facilitate resolution of the optimization problems. However, they are only restricted to the class label, and cannot be applied to other forms of side information such as pairwise similarity/dissimilarity. Moreover, they do not use any regularization scheme to avoid the trace of the kernel matrix from growing unboundedly. For the proposed scg-loss, it not only has linear form, but can be applied to various kinds of side information as well.

Algorithms in the second category exploit the side information via pairwise constraints. The side information can either be provided as class label or pairwise similarity/dissimilarity. The linear constraints are used to ensure that similar data yield small distances and dissimilar data have large distances. If the class labels are provided, we should add one constraint for each paired labeled data. A large number of labels lead to too many constraints. For instance, 1000 labeled data generate 499,500 constraints, which will restrict the algorithms to small-scale problems. The algorithms are accelerated by selecting a small portion of the constraints in [27] since many constraints mutually "overlap" and thus are unnecessary. However, the computational complexity of the algorithms with pairwise constraints are still strongly dependent on the amount of side information [14,18,22].

2.2. Kernel alignment

Given two kernel matrices $K \in \mathbb{R}^{n\times n}$ and $K' \in \mathbb{R}^{n\times n}$, the kernel alignment is defined as

$$\rho(K, K') = \frac{\langle K, K'\rangle_F}{\sqrt{\langle K, K\rangle_F \langle K', K'\rangle_F}}, \tag{1}$$

where $\langle K, K'\rangle_F$ is the inner product of matrices, namely $\langle K, K'\rangle_F = \sum_{ij} K_{ij}K'_{ij}$. Kernel alignment can be viewed as a similarity measurement between two kernel matrices. By the Cauchy-Schwarz inequality $(\sum_{ij} K_{ij}K'_{ij})^2 \le \sum_{ij} K_{ij}^2 \sum_{ij} K_{ij}'^2$, we have $\rho(K, K') \in [-1, 1]$. For binary classification, we set $K' = \mathbf{y}\mathbf{y}^\top$, where $\mathbf{y}$ is a vector of $\{+1, -1\}$ labels, in which case the kernel alignment measures how well the kernel matrix K fits the class labels. We introduce the algorithm of [40], which utilizes the kernel alignment to learn supervised kernels.

Zhu et al. [40] learned a multi-class supervised kernel from the eigenvectors of a normalized Laplacian matrix. We name this algorithm Order Spectral Kernel (OSK). A normalized Laplacian matrix L of a graph is given to characterize the geometric structure of the data, and a target matrix Y defined on labeled data, with entry Y_{ij} set to $+1$ if $y_i = y_j$, and -1 otherwise. Let the eigensystem of L be $\{\lambda_i, \mathbf{v}_i\}$, i.e., $L = \sum_{i=1}^n \lambda_i \mathbf{v}_i \mathbf{v}_i^\top$, where λ_i is sorted in non-decreasing order. Since a smaller eigenvalue of L corresponds to a smoother eigenvector over the graph [31], Zhu et al. impose order constraints to eigenvalues so that the smoother eigenvectors have larger eigenvalues in K. This leads to the following optimization problem:

$$\begin{aligned} \max_K \quad & \rho(K_{\mathrm{tr}}, Y) \\ \text{s.t.} \quad & K = \textstyle\sum_{i=1}^r \mu_i K_i \\ & \mathrm{tr}(K) = 1 \\ & \mu_i \ge 0 \\ & \mu_i \ge \mu_{i+1},\ i = 1, \ldots, r-1, \end{aligned} \tag{2}$$

where $K_i = \mathbf{v}_i\mathbf{v}_i^\top$, K_{tr} is the kernel matrix K restricted on the labeled data, and $\text{tr}(K)$ is the trace of K. Problem (2) can be solved by formulating it as a QCQP. The worst-case complexity of the QCQP using interior-point methods is $\mathcal{O}(r^3)$ per iteration [25]. Note that Y has one negative eigenvalue for multiple classes, thus is no longer a positive semi-definite matrix [27].

2.3. Pairwise constraints

Let $\mathcal{S}$ and $\mathcal{D}$ be the sets of similar and dissimilar data, respectively. The side information forms a convex set $\mathcal{K}$:

$$\mathcal{K} := \{\ K|\ K \succeq 0,\ K_{ii} + K_{jj} - 2K_{ij} \le u,\ \ (\mathbf{x}_i, \mathbf{x}_j) \in \mathcal{S},\ K_{ii} + K_{jj} - 2K_{ij} \ge l,\ \ (\mathbf{x}_i, \mathbf{x}_j) \in \mathcal{D}\ \}, \tag{3}$$

where u and l are the pre-specified constants.

Jain et al. [14] considered the following problem:

$$\min_{K \in \mathcal{K}} D_{ld}(K, K_0), \tag{4}$$

where K_0 was an initial kernel matrix, and $D_{ld}(K, K_0) := \text{tr}(KK_0^{-1}) - \log\det(KK_0^{-1}) - n$, known as the LogDet divergence. We name the algorithm LogDet Kernel (LDK). The above problem is solved by using the Bregman projection algorithm. At each step, we choose one constraint and project the current solution onto this constraint. Assume a similar constraint is picked from $\mathcal{S}$ at one step, then we solve the following problem:

$$\begin{aligned} \min_{K \succeq 0} \quad & D_{ld}(K, K_0) \\ \text{s.t.} \quad & K_{ii} + K_{jj} - 2K_{ij} \le u. \end{aligned} \tag{5}$$

Each constraint projection costs $\mathcal{O}(n^2)$; hence, a single iteration of looping through all constraints costs $\mathcal{O}(cn^2)$, where c is the number of constraints. An important property of LDK is that the learned kernel matrix K can be generalized to a kernel function of the form:

$$k(\mathbf{x}, \mathbf{x}') = k_0(\mathbf{x}, \mathbf{x}') + \sum_{i,j=1}^{n} Q_{ij} k_0(\mathbf{x}_i, \mathbf{x}) k_0(\mathbf{x}_j, \mathbf{x}'), \tag{6}$$

where k_0 is the kernel function corresponding to K_0 and $Q = K_0^{-1}(K - K_0)K_0^{-1}$. Therefore, LDK can be applied to inductive learning.

3. Proposed loss function

Suppose that the side information is provided as pairwise similarity/dissimilarity. We are given a similar set $\mathcal{S}$ and a dissimilar set $\mathcal{D}$. An $n \times n$ matrix T is defined to encode the side information:

$$T_{ij} = \begin{cases} +1, & (\mathbf{x}_i, \mathbf{x}_j) \in \mathcal{S} \\ -1, & (\mathbf{x}_i, \mathbf{x}_j) \in \mathcal{D} \\ 0, & \text{otherwise}. \end{cases} \tag{7}$$

When the class labels are given, they are firstly converted to a set of pairwise constraints, then T is computed accordingly. The matrix T is used to construct a complete graph $\mathcal{G}$ where the data points represent the vertices. In the field of graph theory, a complete graph is a simple undirected graph in which every pair of distinct vertices is connected by a unique edge. The relation between vertices $\mathbf{x}_i$ and $\mathbf{x}_j$ is characterized by the edge weight W_{ij}. All the W_{ij} form an adjacency matrix W, which can represent the graph $\mathcal{G}$.

Previous studies use W to describe the geometric structure of the data and subsequently derive a low-dimensional embedding of the data [1]. Suppose the data are sampled from a smooth manifold $\mathcal{M}$, then the graph $\mathcal{G}$ is a discrete version of $\mathcal{M}$. We consider the following functional to measure the degree of smoothness for a function $f : \mathcal{M} \to \mathbb{R}$:

$$\mathcal{A}(f) = \int_{\mathcal{M}} \|\nabla f\|^2, \tag{8}$$

where ∇f is the gradient vector field of f. If $\mathcal{A}(f)$ is close to zero, we regard f as being "smooth" over the manifold $\mathcal{M}$. We use W_{ij} to characterize the local geometry between the data $\mathbf{x}_i$ and $\mathbf{x}_j$ on the graph $\mathcal{G}$. Then Eq. (8) can be approximated as the following discrete form [2]:

$$\mathcal{A}_{\mathcal{G}}(f_{\mathcal{G}}) = \sum_{i,j=1}^{n} W_{ij}(f_{\mathcal{G}}(\mathbf{x}_i) - f_{\mathcal{G}}(\mathbf{x}_j))^2, \tag{9}$$

where $f_{\mathcal{G}} : X \to \mathbb{R}$. The minimizer $f_{\mathcal{G}}^\star$ of (9) is the smoothest function over $\mathcal{G}$. Eq. (9) is also served as manifold regularization in semi-supervised learning [3].

In this work, we modify Eq. (9) by giving a different definition of W which reflects the side information. In this way, the graph $\mathcal{G}$ characterizes the supervision relations between the data rather than the geometry relations. As we will see later, the new formulation has a close relationship with the kernel alignment. The entries of W are defined as $W_{ij} = \exp(T_{ij})$. A complete graph such as this is referred to as a Supervised Complete Graph (SCG). Using the SCG, we present the following loss function:

$$\text{scg-loss}(K, T) = \sum_{i,j=1}^{n} W_{ij} \| \frac{\phi(\mathbf{x}_i)}{\sqrt{D_{ii}}} - \frac{\phi(\mathbf{x}_j)}{\sqrt{D_{jj}}} \|^2 = \langle K, S \rangle_F, \tag{10}$$

where $K = (K_{ij})_{n\times n}$ is the kernel matrix associated with ϕ, $K_{ij} = \phi(\mathbf{x}_i)^\top \phi(\mathbf{x}_j)$, D is the diagonal matrix with its i-th diagonal entry $D_{ii} = \sum_{j=1}^{n} W_{ji} = \sum_{j=1}^{n} \exp(T_{ji})$, $S = I_n - D^{-1/2} W D^{-1/2}$, and I_n is the $n \times n$ identity matrix. By minimizing (10), we see that ϕ would be smooth over the SCG with respect to the edge weight $\exp(T_{ij})$. In other words, if $\mathbf{x}_i$ and $\mathbf{x}_j$ belong to the same class, the distance between $\phi(\mathbf{x}_i)$ and $\phi(\mathbf{x}_j)$ should be small. However, if $\mathbf{x}_i$ and $\mathbf{x}_j$ are from different classes, the distance between $\phi(\mathbf{x}_i)$ and $\phi(\mathbf{x}_j)$ should not be small. The definition of scg-loss is analogous to the manifold regularization [3], except that now the adjacency matrix represents the side information instead of the geometric structure of the data. Note that S is the normalized Laplacian matrix of SCG; thus, it is a symmetric and positive semi-definite matrix. It can be seen that scg-loss is a linear function of the learned kernel matrix, which allows us to develop efficient algorithms for supervised kernel learning.

4. Analysis

This section presents a theoretical analysis of the relationship between scg-loss and kernel alignment.

4.1. Reformulation of scg-loss

The scg-loss function is rewritten in the following form:

$$\langle K, S \rangle_F = \text{tr}(K) - \langle K, T' \rangle_F, \tag{11}$$

with

$$T'_{ij} = \frac{\exp(T_{ij})}{\sqrt{D_{ii} D_{jj}}}. \tag{12}$$

T' can be seen as a normalized target matrix formed with side information. The normalization step of T' is popular in spectral clustering [26]. We consider the problem of learning a supervised kernel matrix with scg-loss:

$$\begin{aligned} \min_K \quad & \text{tr}(K) - \langle K, T' \rangle_F \\ \text{s.t.} \quad & K \succeq 0 \\ & c_i(K) \le 0, \ 1 \le i \le m, \end{aligned} \tag{13}$$

where $c_i : \mathbb{R}^{n\times n} \to \mathbb{R}$ are constraints. Let $\mathcal{C}$ be the feasible set of problem (13), that is,

$$\mathcal{C} := \{K |\ K \succeq 0, \ c_i(K) \le 0, \ 1 \le i \le m\}. \tag{14}$$

Then we rewrite (13) by switching from minimization to maximization:

$$\begin{aligned} \max_K \quad & \langle K, T' \rangle_F - \text{tr}(K) \\ \text{s.t.} \quad & K \in \mathcal{C}. \end{aligned} \tag{15}$$

It can be seen from (15) that the scg-loss performs regularization in the Tikhonov sense by penalizing the trace of the kernel matrix. This controls the capacity of the search space of possible kernel matrices in order to prevent overfitting and achieve good generalization on test data. Using the equivalence between Tiknonov regularization and Ivanov regularization [13], we have the following theorem:

Theorem 1. *Given the optimization problem (15), there exists a $\tau \ge 0$, such that the following optimization problem*

$$\begin{aligned} \max_K \quad & \langle K, T' \rangle_F \\ \text{s.t.} \quad & \text{tr}(K) \le \tau \\ & K \in \mathcal{C} \end{aligned} \tag{16}$$

is equivalent to (15). By equivalent we mean that if $K^\star$ is a solution of one of the problems, then it is also a solution of the other problem.

Proof. Let $K^\star$ and $K\dagger$ be the solutions of (15) and (16), respectively. We set $\tau = \mathrm{tr}(K^\star)$, then $\tau \geq 0$. We find that

$$\mathrm{tr}(K^\dagger) \leq \tau = \mathrm{tr}(K^\star). \tag{17}$$

Clearly, $K^\star$ and $K\dagger$ are feasible for (16) and (15), respectively.

(i). Assume that $K^\star$ is not the solution of (16), that is, there exists K' satisfying $K' \in \mathcal{C}$ and $\mathrm{tr}(K') \leq \tau$, such that

$$\langle K', T' \rangle_F > \langle K^\star, T' \rangle_F. \tag{18}$$

We also have

$$\mathrm{tr}(K') \leq \tau = \mathrm{tr}(K^\star). \tag{19}$$

By combining (18) and (19), we obtain

$$\langle K', T' \rangle_F - \mathrm{tr}(K') > \langle K^\star, T' \rangle_F - \mathrm{tr}(K^\star). \tag{20}$$

This contradicts the optimality of $K^\star$ in (15). Therefore, $K^\star$ is also the solution of (16).

(ii). Suppose $K^\dagger$ is not the solution of (15). By the optimality of $K^\star$ in (15), we obtain

$$\langle K^\star, T' \rangle_F - \mathrm{tr}(K^\star) > \langle K^\dagger, T' \rangle_F - \mathrm{tr}(K^\dagger). \tag{21}$$

Combining (17) and (21) yields

$$\langle K^\star, T' \rangle_F > \langle K^\dagger, T' \rangle_F, \tag{22}$$

which is a contradiction to the optimality of $K^\dagger$ in (16). Therefore, $K^\dagger$ is also the solution of (15). □

From Theorem 1, we see that minimizing the scg-loss is equivalent to maximizing the inner product between the kernel matrix and target matrix along with regularization in the Ivanov sense.

4.2. Relationship with kernel alignment

If kernel alignment is used, the optimization problem becomes

$$\begin{aligned} &\max_K \ \rho(K, T) \\ &\text{s.t.} \ \ K \in \mathcal{C}. \end{aligned} \tag{23}$$

The kernel alignment is invariant to scales. That is, if we scale K by a positive constant α, the resulting αK will have the same alignment as K. Thus, we can constrain the denominator and maximize the numerator, leading to an equivalent optimization problem of (23):

$$\begin{aligned} &\max_K \ \ \langle K, T'' \rangle_F \\ &\text{s.t.} \ \mathrm{tr}(K^2) \leq 1 \\ &\qquad K \in \mathcal{C}, \end{aligned} \tag{24}$$

where

$$T''_{ij} = \frac{T_{ij}}{\sqrt{\langle T, T \rangle_F}}. \tag{25}$$

Comparing (24) with (16), we find that both the problems maximize the inner product between the kernel matrix and the normalized target matrix, together with regularization by controlling the trace of the kernel matrix. However, the ways in which the target matrix is normalized and the trace is controlled are different. The scg-loss function normalizes the target matrix by dividing the degree of vertices of SCG, whereas for kernel alignment, the target matrix is normalized by dividing the Frobenius norm of T. We note that the denominator in (25) can be removed when we optimize (24). Therefore, the target matrix in (24) is actually treated as unnormalized. In scg-loss, the trace of K is bounded. In contrast, kernel alignment controls the trace of K^2. This difference may appear insignificant, but it in fact affects the computational cost. With the quadratic form of K, the optimization problems are often formulated as QCQP [19,40].

5. Supervised kernel learning with scg-loss

We employ the scg-loss function to develop new kernels that are analogous to OSK and LDK. However, our kernels can be learned more efficiently for a large amount of side information.

5.1. OSK with scg-loss

We learn a supervised kernel from the spectral transformation of a normalized Laplacian matrix, which is similar to OSK. Replacing the kernel alignment with the scg-loss in (2), we derive the following problem:

$$\begin{aligned} \min_{\mu_1,\ldots,\mu_r} \quad & \textstyle\sum_{i=1}^{r} \operatorname{tr}(K_i S)\mu_i \\ \text{s.t.} \quad & \mu_i \geq 0 \\ & \textstyle\sum_{i=1}^{r} \mu_i = 1 \\ & \mu_i \geq \mu_{i+1},\ i = 1,\ldots,r-1. \end{aligned} \tag{26}$$

The transformation between problems (2) and (26) is given in Appendix A. The above problem is an LP, which, in the worst case, is solved in $\mathcal{O}(r^{2.5})$ operations per iteration and requires a total of $\mathcal{O}(r^{3.5})$ complexity according to Karmarkar's algorithm [17], which is more efficient than QCQP.

5.2. LDK with scg-loss

An efficient variant of LDK is derived by replacing the pairwise constraints with the scg-loss in problem (4), leading to the following optimization problem:

$$\min_{K\succeq 0} D_{ld}(K, K_0) + \gamma \operatorname{tr}(KS), \tag{27}$$

where $\gamma > 0$. We solve the problem by ignoring the positive semi-definiteness constraint and set the derivative with respect to K to zero:

$$K_0^{-1} - K^{-1} + \gamma S = 0. \tag{28}$$

Then we obtain a closed-form solution

$$K = (K_0^{-1} + \gamma S)^{-1}. \tag{29}$$

We see that the solution K in (29) is positive definite because of the positive definiteness of K_0 and S. The computational complexity of (29) is $\mathcal{O}(n^3)$. Compared with problem (4), the problem (27) is solved more efficiently when a large amount of side information needs to be processed. More importantly, its computational complexity is independent of the amount of side information.

Computation of the out-of-sample data requires us to extend the learned kernel matrix K to a kernel function. To this end, we have the following theorem:

Theorem 2. *We are given the optimization problem (27). Then for each $\gamma > 0$, there exists a unique $\tau \geq 0$, such that the optimization problem*

$$\min_{K\succeq 0} D_{ld}(K, K_0) \quad \text{s.t. } \operatorname{tr}(KS) \leq \tau \tag{30}$$

is equivalent to (27). By equivalent we mean that if $K^\star$ is a solution of one of the problems, then it is also a solution of the other problem.

Theorem 2 shows that Tikhonov regularization can be switched to Ivanov regularization and vice versa. The proof of this theorem is similar to that of Theorem 1, thus we omit it here. We find that the form of optimization problem (30) has been studied in [14]. More specifically, let the initial kernel function be $k_0(\mathbf{x}, \mathbf{x}') = \phi(\mathbf{x})^\top \phi(\mathbf{x}')$, which generates K_0 upon the training data, and the solution to problem (30) be $K^\star$. Using theorem 4 in [14], the kernel matrix $K^\star$ can be extended to a kernel function of the form:

$$k(\mathbf{x}, \mathbf{x}') = \phi(\mathbf{x})^\top U \phi(\mathbf{x}'), \tag{31}$$

where $U = I_n + \Phi^\top Q \Phi$, $Q = K_0^{-1}(K^\star - K_0)K_0^{-1}$ and $\Phi = [\phi(\mathbf{x}_1), \ldots, \phi(\mathbf{x}_n)]$. The kernel function (31) can be further written as:

$$k(\mathbf{x}, \mathbf{x}') = k_0(\mathbf{x}, \mathbf{x}') + \sum_{i,j=1}^{n} Q_{ij} k_0(\mathbf{x}_i, \mathbf{x}) k_0(\mathbf{x}_j, \mathbf{x}'), \tag{32}$$

Then kernel function k can be used to compute the out-of-sample extensions.

6. Experimental results

We consider the tasks of dimensionality reduction and classification. For dimensionality reduction, we compared SCG-LDK with Principal Component Analysis (PCA) [16], Kernel PCA [30], Laplacian Eigenmap [1], and Colored Maximum Variance Unfolding (CMVU) [32]. For classification, our SCG-based algorithms (SCG-OSK and SCG-LDK) are compared with the state-of-the-art kernel learning algorithms: OSK [40], LDK [14], Fisher Score Kernel (FSK) [21], Simple Non-Parametric Kernel

Table 2
Description of the data sets.

Data set	#data	#class	#feature
Protein	116	6	20
Iris	150	3	4
Wine	178	3	13
Sonar	208	2	60
Glass	214	6	9
Heart	270	2	13
Liver	345	2	6
Ionosphere	351	2	34
Breast	569	2	30
Balance	625	3	4
Adult-a1a	1605	2	14
Adult-a2a	2265	2	14
Adult-a3a	3185	2	14
Adult-a4a	4781	2	14
Adult-a5a	6414	2	14
Adult-a6a	11220	2	14
USPS	4000	10	256

(SimpleNPK) [41], and Ideal Regularized Gaussian (IR-Gaussian) [27]. The Gaussian kernel is chosen as a baseline. Our aim is to determine the effectiveness and efficiency of the scg-loss. One synthetic example and 17 real data sets are used for evaluation. Among the real data sets, one is a handwritten digit data set from the US Postal Service (USPS), whereas the others are from the UCI machine learning repositories, shown in Table 2. The USPS data set contains 10 digits and each digit has 400 images. The Adult data set is used for large-scale learning [15]. The goal of this data set is to predict whether a household has an income greater than $50,000.

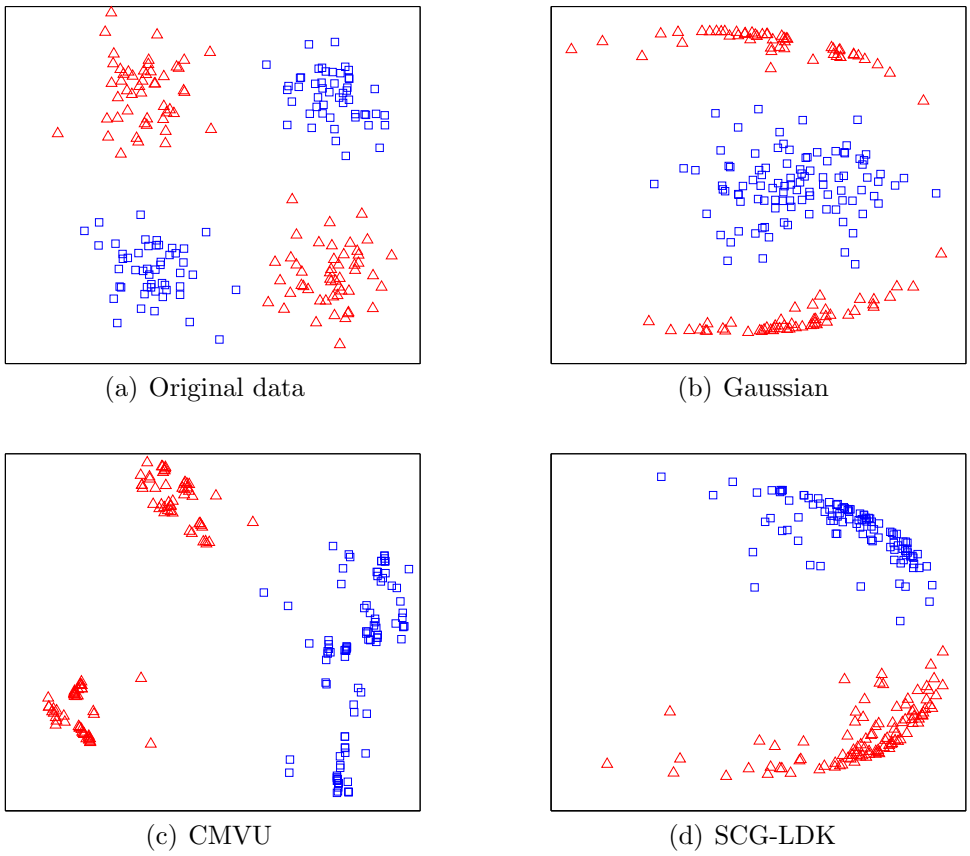
The normalized graph Laplacian is constructed with {0, 1} weights and 5 nearest neighbors, except for the adult data sets where 20 nearest neighbors are used. The accuracy is computed using the LIBSVM toolbox [6], where the trade-off parameter C is tuned via five-fold cross validation over the set {0.1, 1, 10, 100, 1000}. The width of the Gaussian kernel is also adjusted via cross validation. The "one-against-one" strategy is applied to solve the multi-class problem. In effect, we train $k(k-1)/2$ classifiers in which k is the number of classes. Each classifier uses the training data from two different classes. A voting approach is employed for classification. Each binary classification is considered to be a voting session where votes can be cast for all data points. In the end, the point is assigned to the class with the maximum number of votes. We record the running time of the algorithms. This time includes the computation of the kernel matrix and the elapsed time for solving the optimization problem. The results are averaged over 20 runs. The Friedman test is applied for statistical testing at the 0.05 significance level [9]. The null hypothesis is that all algorithms perform equivalently on average. If the null hypothesis is rejected, the Nemenyi test is applied to determine which algorithms perform differently [24]. All experiments are conducted on a 64-bit Windows PC with 3.2 GHz CPU and 32GB RAM, and implemented in MATLAB. The parameter γ of SCG-LDK is also tuned using cross validation over the set {0.01, 0.1, 1, 10, 100}. The QCQP and LP are solved using the SeDuMi toolbox [33].

6.1. Dimensionality reduction

The experiments were conducted on a synthetic example and the USPS data set. The setting is similar to [32] in which all the labels are utilized. For the sake of visualization, we perform eigendecomposition on the kernel matrix K and use two top eigenvalues associated with the corresponding eigenvectors for embedding [10]. This is the embedding yielded by kernel PCA using the kernel matrix K.

We begin with the synthetic example shown in Fig. 1(a). The data are divided into two classes (shown by different symbols and colors). Each class contains 100 points and the distribution of each class is a bimodal Gaussian distribution. Note that these two classes are not linearly separated. We use Gaussian, CMVU and SCG-LDK for embedding, shown in Fig. 1(b)–(d). The Gaussian finds one class in a cluster, but separates the other class into two parts. Both CMVU and SCG-LDK separate the points from different classes. However, CMVU yields two subclasses for one class. It appears to be three clusters by inspection. SCG-LDK yields a more meaningful embedding which represents two well-separated clusters.

Then, we choose the USPS data sets for embedding, which is visualized in a two-dimensional space, shown in Fig. 2. Different colors represent different classes. The color-digit correspondence is given at the bottom of Fig. 2. As shown in the figures, unsupervised methods, such as PCA and Laplacian Eigenmap, yield an embedding of different classes mixed together. CMVU produces a clearer embedding, but there are still large overlaps between different classes. In contrast, SCG-LDK produces good visualizations. We observe that digits 3, 6, 8, and 0 are well separated. There are some overlaps between digits 2 and 5. Large overlaps are found for digits 4 and 9, 1 and 7, respectively. This indicates high similarity between these paired digits.



(a) Original data (b) Gaussian

(c) CMVU (d) SCG-LDK

Fig. 1. Synthetic example of bimodal Gaussian distributions.

6.2. Comparison on small-scale data sets

We evaluate the algorithms on ten UCI data sets by comparing the accuracy and running time. The data sets are split into 70% labeled data and 30% test data. The average rank of each algorithm for the Friedman test is computed. The results are shown in Table 3. We did not record the running time of the Gaussian function since there is no kernel-learning algorithm for a Gaussian. In terms of the average rank, the best algorithm is SCG-LDK with rank 1.95, followed by LDK and IR-Gaussian, with rank 3.45 and 3.50, respectively. SCG-OSK and OSK perform the worst, with a rank of 6.20 and 6.10, respectively. Since SCG-OSK and OSK are designed for the data sampled from a smooth manifold, they may not be suitable for these UCI data sets. We note that the performance of SCG-OSK and OSK is almost indistinguishable. For statistical comparisons of the algorithms, we use the Friedman test with the corresponding post-hoc tests. The Friedman statistic is distributed according to the F-distribution with $8-1=7$ and $(8-1)\times(10-1)=63$ degrees of freedom. The value of the Friedman statistic is 5.17, and the critical value of $F(7, 63)$ is 2.16 at the 0.05 significance level. Thus, the null hypothesis is rejected. Then we applied the Nemenyi test for pairwise comparisons. The critical difference is 3.32. Since $5.35-1.95=3.40>3.32$, we find that SCG-LDK performs significantly more accurately than Gaussian, OSK, and SCG-OSK. We did not detect any significant differences between the other algorithms.

From the point of view of efficiency, SCG-OSK is faster than OSK. On balance, SCG-OSK roughly gains a factor of 30 in computational complexity over OSK. The extent of acceleration can be seen from the computational complexity. SCG-OSK scales $\mathcal{O}(r^{3.5})$, whereas OSK scales $\mathcal{O}(r^3t)$ in which t is the number of iterations. SCG-LDK is also more efficient than LDK. SCG-LDK is about 800 times faster than LDK on breast. Comparing the $\mathcal{O}(n^3)$ complexity of OSK-LDK with $\mathcal{O}(cn^2t)$ complexity of LDK, where c is the number of constraints and t is the number of iterations, LDK is time-consuming when processing a large amount of side information.

We depict the relation between the CPU time and the number of labels by presenting an experiment on Heart by varying the ratio of available labels to total data from 0.05 to 1 with 20 equally spaced values. The results are plotted in Fig. 3. The CPU time of the second subfigure is plotted in logarithmic scale. The figures show that the running time of LDK increases quadratically because the number of constraints is quadratic in the number of labels. The computational cost of OSK increases almost linearly as more labels are available. Since the computational complexities of SCG-OSK and OSK-LDK

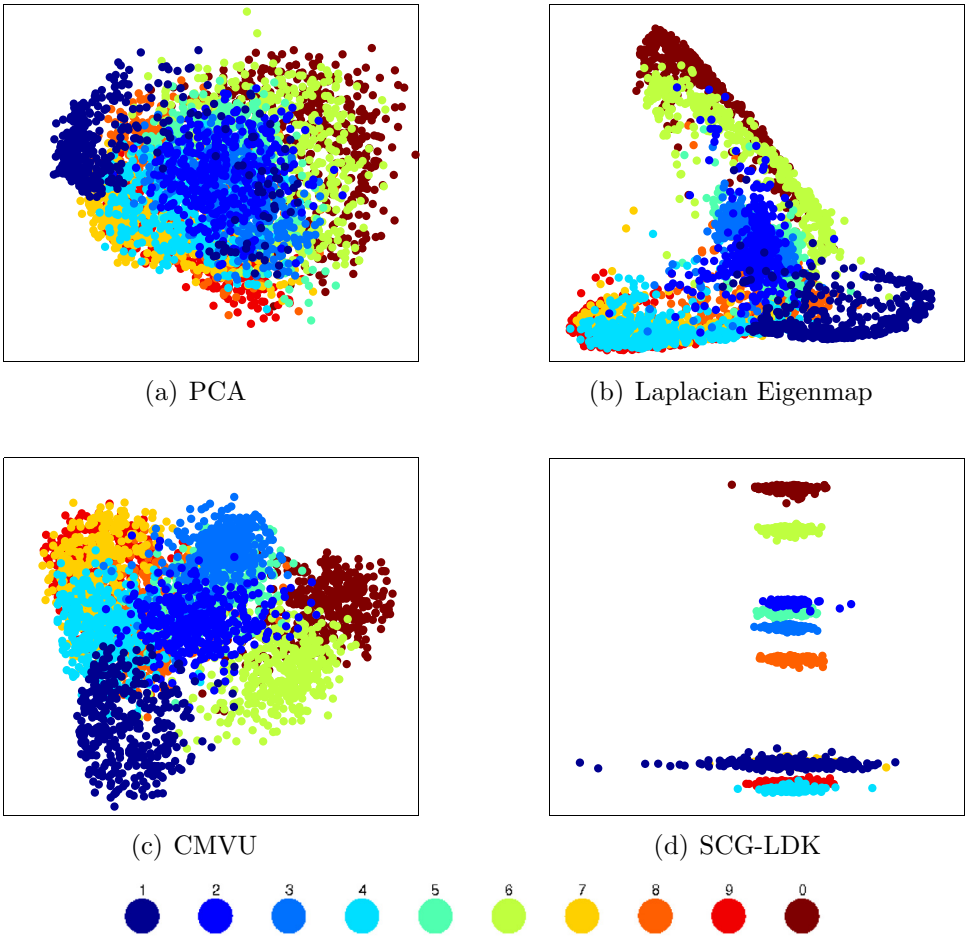
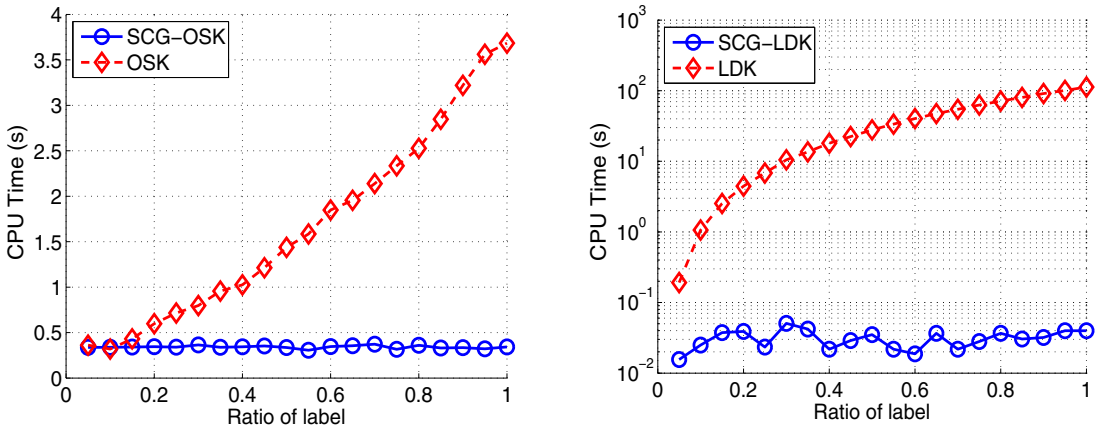


Fig. 2. Embeddings of USPS data set.

Table 3
Accuracy (%) and running time (second) on ten UCI data sets. Each cell has two rows: the upper row shows the accuracy and standard deviation; the lower row reports the running time. The best accuracy is presented in bold. The last row provides the average rank of each algorithm.

Data set	Gaussian	FSK	SimpleNPK	IR-Gaussian	OSK	LDK	SCG-OSK	SCG-LDK
Protein	70.2 ± 7.1	79.1 ± 6.4	73.9 ± 9.5	73.2 ± 6.3	70.0 ± 6.0	74.7 ± 6.6	71.1 ± 7.2	**75.3** ± 7.0
	–	0.32	0.06	0.06	3.09	0.55	0.61	0.01
Wine	95.7 ± 2.0	**96.7** ± 1.9	95.3 ± 2.0	96.4 ± 2.4	95.3 ± 1.6	96.6 ± 1.8	95.7 ± 2.3	**96.7** ± 1.7
	–	0.53	0.12	0.6	3.55	0.78	0.49	0.01
Iris	95.3 ± 2.6	94.4 ± 3.2	96.9 ± 2.8	95.3 ± 2.2	97.6 ± 2.7	96.2 ± 2.1	**97.8** ± 1.8	94.9 ± 4.1
	–	0.38	0.06	0.06	1.37	0.69	0.27	0.01
Sonar	78.1 ± 4.5	86.3 ± 2.6	81.3 ± 3.8	85.8 ± 4.7	80.3 ± 4.7	85.9 ± 3.4	85.2 ± 3.6	**86.4** ± 2.8
	–	0.27	0.12	0.06	3.93	3.06	0.50	0.01
Glass	56.0 ± 3.7	58.1 ± 3.2	52.1 ± 5.4	57.9 ± 6.0	56.4 ± 5.9	59.7 ± 3.7	55.0 ± 4.5	**61.3** ± **2.6**
	–	0.60	0.12	0.12	2.64	0.41	0.47	0.01
Heart	82.3 ± 3.2	76.3 ± 3.5	85.5 ± 3.9	**85.9** ± 3.1	66.3 ± 3.5	76.8 ± 4.2	65.5 ± 4.1	85.6 ± 3.5
	–	0.35	0.19	0.12	3.62	11.03	0.53	0.06
Liver	66.8 ± 3.5	64.4 ± 3.5	67.3 ± 6.2	**71.5** ± 6.6	66.8 ± 8.3	67.2 ± 4.5	66.0 ± 6.8	69.0 ± 5.6
	–	0.57	0.31	0.30	4.60	25.90	0.45	0.06
Ionosphere	94.8 ± 1.5	93.0 ± 1.7	87.9 ± 6.4	95.7 ± 1.5	89.1 ± 3.6	95.7 ± 3.0	87.9 ± 3.5	**97.0** ± 1.3
	–	0.79	0.31	0.31	4.74	28.19	0.75	0.12
Breast	95.2 ± 1.8	95.1 ± 1.4	92.7 ± 3.0	96.1 ± 1.0	94.5 ± 1.7	96.6 ± 1.6	94.4 ± 1.7	**97.5** ± 1.1
	–	0.89	0.89	0.84	10.08	252.14	0.81	0.31
Balance	93.5 ± 1.9	94.6 ± 1.4	95.1 ± 0.9	92.7 ± 2.0	91.2 ± 1.4	91.6 ± 1.5	90.3 ± 3.1	**96.0** ± 1.5
	–	1.23	1.20	1.36	15.68	104.91	0.54	0.37
Average rank	5.35	4.25	5.20	3.50	6.10	3.45	6.20	1.95



(a) SCG-OSK vs. OSK (b) SCG-LDK vs. LDK

Fig. 3. Plot of CPU time vs. number of labels on heart data set. The x-axis represents the ratio of available labels to total data.

Table 4
Accuracy (%) and standard deviation of unlabeled and test data for SCG-LDK. The average accuracy over all the data sets is shown. The Wilcoxon signed-rank test is given in the last row. See the text for the explanation of p and h.

	Unlabel	Test
Protein	68.8 ± 7.4	71.3 ± 5.8
Wine	96.0 ± 3.7	98.0 ± 1.4
Iris	94.7 ± 3.0	95.7 ± 3.0
Sonar	80.2 ± 5.6	79.5 ± 4.6
Glass	61.5 ± 4.9	66.0 ± 6.7
Heart	82.3 ± 4.4	81.7 ± 2.8
Liver	62.5 ± 5.9	64.6 ± 4.3
Ionosphere	96.1 ± 2.2	93.4 ± 1.7
Breast	97.3 ± 1.3	96.7 ± 1.3
Balance	93.5 ± 2.3	90.4 ± 2.8
Average	83.3	83.7
[p, h]	[0.6785,0]	

are independent of the number of labels, the running times of these methods are nearly invariant as the number of labels changes. This would be advantageous when a large amount of side information is available.

6.3. Comparison on out-of-sample extension

We conduct an experiment to evaluate the out-of-sample performance for SCG-LDK. Eq. (32) is used to compute the out-of-sample extensions. The data sets are split into 40% labeled data, 30% unlabeled data, and 30% test data. The training set comprises the labeled and unlabeled data. The accuracy is computed for both the unlabeled and test data, shown in Table 4. We observe that the out-of-sample performance is comparable to its in-sample performance in most cases. We also compute the average accuracy over all the data sets. The average accuracy of the unlabeled data closely approximates that of the test data. The last row in Table 4 shows the Wilcoxon signed-rank test, where p is the p-value, and $h = 0$ indicates the average difference between the accuracy of the unlabeled and test data is not significantly different from zero at the 0.05 significance level, $h = 1$ indicates the average difference is significantly different from zero. The statistical test indicates that SCG-LDK extends to out-of-sample data relatively well. We conjecture the underlying reasons. The kernel function defined in (32) is a smooth function when the initial kernel function is chosen as the Gaussian kernel. It recovers the kernel matrix (29) exactly using the training data, which implies that the kernel function (32) performs the same as the kernel matrix (29) upon the training data. The performance of unseen data would be expected to be close to that of the training data when the kernel function extends smoothly to the unseen data.

Table 5
Running time (s) and accuracy (%) on adult data set. Each cell has two rows: the upper row shows the accuracy and standard deviation; the lower row reports the running time. The best accuracy is presented in bold. The last row provides the average rank of each algorithm.

Data set	# Label	Gaussian	FSK	SimpleNPK	IR-Gaussian	SCG-OSK	SCG-LDK
Adult-a1a	200	68.7 ± 1.8	74.0 ± 3.8	70.5 ± 12.1	76.6 ± 3.2	67.9 ± 14.4	**78.3** ± 4.9
		–	10.81	8.12	15.93	6.77	2.24
Adult-a2a	400	70.1 ± 1.8	74.5 ± 2.4	72.7 ± 7.4	76.6 ± 2.4	70.6 ± 5.2	**79.1** ± 2.4
		–	29.81	22.11	43.96	11.45	4.70
Adult-a3a	600	69.9 ± 1.5	75.7 ± 2.3	70.1 ± 7.3	77.7 ± 1.4	69.1 ± 5.8	**81.2** ± 2.1
		–	64.19	58.60	117.29	27.08	11.10
Adult-a4a	800	70.7 ± 1.3	75.8 ± 1.9	71.1 ± 4.7	79.4 ± 1.6	72.0 ± 5.3	**82.4** ± 0.8
		–	119.27	181.91	372.64	77.81	33.95
Adult-a5a	1000	71.0 ± 1.1	76.6 ± 1.3	72.0 ± 6.6	79.8 ± 1.3	71.9 ± 7.0	**81.2** ± 2.3
		–	288.02	418.29	928.34	172.41	86.73
Adult-a6a	1200	70.9 ± 1.6	76.0 ± 0.8	73.2 ± 2.1	80.7 ± 0.7	69.2 ± 5.8	**81.4** ± 1.9
		–	502.19	1994.01	4575.80	837.26	375.67
average rank		5.50	3.00	4.17	2.00	5.33	1.00

6.4. Comparisons on large-scale data sets

The Gaussian, FSK, SimpleNPK, IR-Gaussian, SCG-OSK, and SCG-LDK are chosen for evaluation on the adult data sets. We aim to determine the extent to which the computational costs change when additional data and labels are available. The Friedman test is applied for the statistical comparisons of accuracy. The running time as well as accuracy are recorded in Table 5. SCG-LDK performs the best, with rank 1. The second best algorithm is IR-Gaussian with rank 2. Gaussian performs the worst with the highest rank. The Friedman statistic is 87.65, which is larger than the critical value $F(5, 25) = 2.60$. Thus, we reject the null hypothesis. Then the Nemenyi test is used for pairwise comparisons. The critical difference is 3.08. It is found that SCG-LDK performs significantly more accurately than Gaussian, SimpleNPK, and SCG-OSK, whereas IR-Gaussian is significantly more accurate than Gaussian and SCG-OSK. SCG-LDK is also efficient. It has low computational costs and is about 2–12 times faster than other algorithms, except for Gaussian.

7. Conclusions

This paper proposed a novel loss function for learning supervised kernels. We constructed a complete graph for which the edge weights are computed with side information. Our loss function is proposed by requiring the learned kernel to be smooth over the complete graph. The theoretical analysis showed that the proposed loss function aims to maximize the inner product between the learned kernel and target matrix derived from the side information, along with regularization by controlling the trace of the learned kernel. As a linear form of the learned kernel matrix, our loss function was shown to use the side information efficiently. We used the proposed loss function to develop efficient supervised kernel learning algorithms. The computational complexity of our algorithms is independent of the amount of side information. Empirical results indicated that the proposed loss function exploits the side information efficiently and effectively. In future research, we plan to employ our loss function to develop additional supervised kernel learning algorithms.

Acknowledgment

This paper is partially supported by National Natural Science Foundation of China (11526145, 61272252) and Natural Science Foundation of Guangdong Province (2015A030313544).

Appendix A

We show how to transform the problem (2) into (26). Replacing the kernel alignment in (2) with scg-loss, we obtain the objective function as

$$\begin{aligned}\langle K, S\rangle_F &= \operatorname{tr}(KS)\\ &= \operatorname{tr}\left[\left(\sum_{i=1}^{r}\mu_i K_i\right)S\right]\\ &= \sum_{i=1}^{r}\operatorname{tr}(K_i S)\mu_i.\end{aligned}$$

The left side of the constraint $\operatorname{tr}(K) = 1$ is converted into the following:

$$\operatorname{tr}(K) = \operatorname{tr}\left(\sum_{i=1}^{r}\mu_i K_i\right)$$

$$= \mathrm{tr}\left(\sum_{i=1}^{r} \mu_i \mathbf{v}_i \mathbf{v}_i^\top\right)$$
$$= \sum_{i=1}^{r} \mu_i \mathrm{tr}(\mathbf{v}_i^\top \mathbf{v}_i)$$
$$= \sum_{i=1}^{r} \mu_i.$$

The last equation originates from the normalization of $\mathbf{v}_i$. Then we derive the problem (26).

References

[1] M. Belkin, P. Niyogi, Laplacian eigenmaps for dimensionality reduction and data representation, Neural Comput. 15 (6) (2003) 1373–1396.
[2] M. Belkin, P. Niyogi, Using manifold structure for partially labeled classification, in: Advances in Neural Information Processing Systems, 15, 2003, pp. 929–936.
[3] M. Belkin, P. Niyogi, V. Sindhwani, Manifold regularization: a geometric framework for learning from labeled and unlabeled examples, J. Mach. Learn. Res. 7 (2006) 2399–2434.
[4] S. Bucak, R. Jin, A. Jain, Multiple kernel learning for visual object recognition: a review, IEEE Trans. Pattern Anal. Mach. Intell. 36 (7) (2014) 1354–1369.
[5] G. Cawley, N. Talbot, Kernel learning at the first level of inference, Neural Netw. 53 (2014) 69–80.
[6] C. Chang, C. Lin, LIBSVM : a library for support vector machines, ACM Trans. Intell. Syst. Technol. 2 (3) (2007) 389–396.
[7] W. Chen, Y. Shao, N. Hong, Laplacian smooth twin support vector machine for semi-supervised classification, Int. J. Mach. Learn. Cybern. 5 (3) (2014) 459–468.
[8] N. Cristianini, J. Shawe-Taylor, A. Elisseeff, J. Kandola, On kernel-target alignment, in: Advances in Neural Information Processing Systems, 15, 2002, pp. 367–373.
[9] J. Demšar, Statistical comparisons of classifiers over multiple data sets, J. Mach. Learn. Res. 7 (2006) 1–30.
[10] J. Ham, D. Lee, S. Mika, B. Schölkopf, A kernel view of the dimensionality reduction of manifolds, in: Proceedings of the 21st Annual International Conference on Machine Learning, 2004, pp. 47–54.
[11] S. Hoi, R. Jin, M. Lyu, Learning nonparametric kernel matrices from pairwise constraints, in: International Conference on Machine Learning, 2007, pp. 361–368.
[12] S. Huang, L. Jin, K. Xue, Y. Fang, Online primal-dual learning for a data-dependent multi-kernel combination model with multiclass visual categorization applications, Inf. Sci. 320 (2015) 75–100.
[13] V. Ivanov, V. Vasin, V. Tanana, Theory of linear ill-posed problems and its applications, Inverse and Ill-posed Problems Series, Brill Publishers, Leiden, Netherlands, 2002.
[14] P. Jain, B. Kulis, J.V. Davis, I.S. Dhillon, Metric and kernel learning using a linear transformation, J. Mach. Learn. Res. 13 (2012) 519–547.
[15] T. Joachims, Making large-scale SVM learning practical, Adv. Kernel Methods Support Vector Learn. (1999) 169–184.
[16] I. Jolliffe, Principal component analysis, 2nd edition, Springer, 2002.
[17] N. Karmarkar, A new polynomial time algorithm for linear programming, Combinatorica 4 (4) (1984) 373–395.
[18] B. Kulis, M. Sustik, I. Dhillon, Low-rank kernel learning with Bregman matrix divergences, J. Mach. Learn. Res. 10 (2009) 341–376.
[19] G. Lanckriet, N. Cristianini, P. Bartlett, L. Ghaoui, M. Jordan, Learning the kernel matrix with semidefinite programming, J. Mach. Learn. Res. 5 (2004) 27–72.
[20] F. Li, Y. Fu, Y. Dai, C. Sminchisescu, J. Wang, Kernel learning by unconstrained optimization, in: International Conference on Artificial Intelligence and Statistics, 2009, pp. 328–335.
[21] J.-B. Li, Y.-H. Wang, S.-C. Chu, J. Roddick, Kernel self-optimization learning for kernel-based feature extraction and recognition, Inf. Sci. 257 (2014) 70–80.
[22] Z. Lu, P. Jain, I.S. Dhillon, Geometry-aware metric learning, in: International Conference on Machine Learning, 2009, pp. 673–680.
[23] A. Nazarpour, P. Adibi, Two-stage multiple kernel learning for supervised dimensionality reduction, Pattern Recognit. 48 (2015) 1854–1862.
[24] P. Nemenyi, Distribution-Free Multiple Comparisons, Princeton University, 1963 Ph.D. thesis.
[25] Y. Nesterov, A. Nemirovskii, Interior Point Polynomial Methods in Convex Programming, Society for Industrial and Applied Mathematics, Philadelphia, PA, 1994.
[26] A.Y. Ng, M.I. Jordan, Y. Weiss, On spectral clustering: analysis and an algorithm, in: Advances in Neural Information Processing Systems, 14, 2002, pp. 849–856.
[27] B. Pan, J. Lai, L. Shen, Ideal regularization for learning kernels from labels, Neural Netw. 56 (2014) 22–34.
[28] B. Pan, J.J. Xia, P. Yuan, J. Gateno, H.H.S. Ip, Q. He, P.K.M. Lee, B. Chow, X. Zhou, Incremental kernel ridge regression for the prediction of soft tissue deformations, in: The 15th International Conference on Medical Image Computing and Computer Assisted Intervention, Nice, France, 2012, pp. 99–106.
[29] X. Peng, D. Chen, L. Kong, D. Xu, Interval twin support vector regression algorithm for interval input-output data, Int. J. Mach. Learn. Cybern. 6 (5) (2015) 719–732.
[30] B. Schölkopf, A. Smola, K. Müller, Nonlinear component analysis as a kernel eigenvalue problem, Neural Comput. 10 (5) (1998) 1299–1319.
[31] A. Smola, R.I. Kondor, Kernels and regularization on graphs, in: Learning Theory and Kernel Machines, Springer, 2003, pp. 144–158.
[32] L. Song, A. Smola, K. Borgwardt, A. Gretton, Colored maximum variance unfolding, in: Advances in Neural Information Processing Systems, 21, 2008, pp. 1385–1392.
[33] J. Sturm, Using SeDuMi 1.02, a MATLAB toolbox for optimization over symmetric cones, Optim. Methods Softw. 10–11 (1999) 625–653.
[34] M. Tanveer, Newton method for implicit Lagrangian twin support vector machines, Int. J. Mach. Learn. Cybern. 6 (6) (2015) 1029–1040.
[35] M. Tian, W. Wang, An efficient gaussian kernel optimization based on centered kernel polarization criterion, Inf. Sci. 322 (2015) 133–149.
[36] H. Xiong, M. Swamy, M. Ahmad, Optimizing the kernel in the empirical feature space, IEEE Trans. Neural Netw. 16 (2) (2005) 460–474.
[37] M. Yang, L. Zhang, S.-K. Shiu, D. Zhang, Robust kernel representation with statistical local features for face recognition, IEEE Trans. Neural Netw. Learn. Syst. 24 (6) (2013) 900–912.
[38] X. Zhang, M. Mahoor, Task-dependent multi-task multiple kernel learning for facial action unit detection, Pattern Recognit. 51 (2016) 187–196.
[39] S. Zhong, T. Chen, F. He, Y. Niu, Fast Gaussian kernel learning for classification tasks based on specially structured global optimization, Neural Netw. 57 (2014) 51–62.
[40] X. Zhu, J. Kandola, Z. Ghahramani, J. Lafferty, Nonparametric transforms of graph kernels for semi-supervised learning., in: Advances in Neural Information Processing Systems, 17, 2004, pp. 1641–1648.
[41] J. Zhuang, I.W. Tsang, S.C.H. Hoi, A family of simple non-parametric kernel learning algorithms, J. Mach. Learn. Res. 12 (2011) 1313–1347.