Contents lists available at ScienceDirect

Neurocomputing

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

Enhancing semi-supervised learning through label-aware base kernels

Qiaojun Wang [a,*], Kai Zhang [b], Zhengzhang Chen [b], Dequan Wang [c], Guofei Jiang [b], Ivan Marsic [a]

[a] Department of Electrical and Computer Engineering, Rutgers University, Piscataway, NJ, United States
[b] NEC Laboratories America, Inc., 4 Independence Way, Princeton, NJ, United States
[c] School of Computer Science, Fudan University, Shanghai, China

ARTICLE INFO

Article history:
Received 4 March 2015
Received in revised form
14 June 2015
Accepted 25 July 2015
Communicated by Haowei Liu

Keywords:
Semi-supervised learning
Multiple-kernel learning
Kernel target alignment
Label-aware base kernels

ABSTRACT

Currently, a large family of kernel design methods for semi-supervised learning (SSL) problems builds the kernel by weighted averaging of predefined base kernels (i.e., those spanned by kernel eigenvectors). Optimization of the base kernel weights has been studied extensively in the literature. However, little attention was devoted to designing high-quality base kernels. The eigenvectors of the kernel matrix, which are computed irrespective of class labels, may not always reveal useful structures of the target. As a result, the generalization performance can be poor however hard the base kernel weighting is tuned. On the other hand, there are many SSL algorithms whose focus are not on kernel design but on the estimation of the class labels directly. Motivated by the label propagation approach, in this paper we propose to construct novel kernel eigenvectors by injecting the class label information under the framework of eigenfunction extrapolation. A set of "label-aware" base kernels can be obtained with greatly improved quality, which leads to higher target alignment and henceforth better performance. Our approach is computationally efficient, and demonstrates encouraging performance in semi-supervised classification and regression tasks.

1. Introduction

Semi-supervised learning (SSL) is a useful learning paradigm that can make use of unlabeled samples to boost the learning performance with only limited supervision. Among various directions that have been pursued, for example, graph based algorithms [1,2], low-density separation [3], transductive SVM [4–6], Semi-Definite Programming (SDP) [7], ensemble method [8], high order [9], semi-supervised kernel design turns to be a promising one because it allows the abundant theories and algorithms in kernel methods to be adopted directly in solving SSL problems. In particular, a large family of algorithms for semi-supervised kernel relies on spectral transformation, where the eigenvectors of the kernel matrix (or the graph Laplacian) are used together with the rectified eigenvalues to build the new kernel.

Lots of empirical successes have been observed with the family of semi-supervised kernels based on spectral transforms. However, there are still some concerns with them. First, building a kernel solely based on rectifying the kernel eigen-spectrum may be restrictive in terms of acquiring desired kernel. Note that eigenvectors of the empirical kernel matrix (or graph Laplacian) are computed in an unsupervised manner, entirely irrespective of the class labels. They can be inaccurate due to various practical factors such as noise, kernel types or parameters, or class separability. Therefore, these eigenvectors may not reveal useful structures for classification, and the base kernels they span can have low alignment with the target, while the alignment of the mixed kernel depends crucially on the alignment of the individual base kernels. Second, the optimization procedure involved can be quite expensive. For example, computing the eigenvalue decomposition of the Laplacian already takes $O(n^3)$ time and $O(n^2)$ memory. The time complexity of Quadratically constrained quadratic program (QCQP) ($O(n^4)$) [10] and SDP ($O(n^{4.5})$) [7] is also quite demanding.

To solve these problems, we propose a new way for designing base kernels used in semi-supervised kernel learning. Besides using the eigenvectors from the original kernel matrix or graph Laplacian, we also compute a new set of more "accurate" eigenvectors that are expected to be better aligned to the target. Our key observation is that the kernel eigenvectors and class labels have some intrinsic connections. In particular, the ideal kernel eigenvectors are deemed equivalent as the class labels. Inspired by this, we compute a set of desired kernel eigenvectors by extrapolating the ideal kernel eigenfunction. Such extrapolation builds upon important proximity structures encoded in the input patterns.

* Corresponding author.
E-mail addresses: qjwang@rutgers.edu (Q. Wang), kzhang@nec-labs.com (K. Zhang), zchen@nec-labs.com (Z. Chen), dqwang12@fudan.edu.cn (D. Wang), gfi@nec-labs.com (G. Jiang), marsic@rutgers.edu (I. Marsic).

http://dx.doi.org/10.1016/j.neucom.2015.07.072
0925-2312/© 2015 Elsevier B.V. All rights reserved.

More importantly, it directly incorporates class labels in the computation. Therefore, the label-aware eigenvectors are empirically more aligned to the target compared with the unsupervised kernel eigenvectors. This directly leads to a set of base kernels with higher quality, and the overall alignment of the mixed kernel will also be improved with better generalization performance. In addition, we use low-rank approximation to compute useful eigenvectors from the original kernel matrix, therefore our approach is computationally very efficient and only requires linear time and space complexities.

It is worthwhile to note that the main contribution of this paper is to explore new ways of constructing base kernels, instead of how to combine the base kernels, in semi-supervised kernel learning. The latter has been studied extensively in the literature as has been discussed [7,11–14], and therefore will not be the focus of this paper. Of course, in order to evaluate the usefulness of the newly proposed base kernels against traditional base kernels, we will still resort to existing methods of kernel combination so as to finally obtain a mixed kernel and its testing accuracy. The main contribution of this paper is as follows. First, we propose to use class labels to improve the quality of base kernels, via the framework of eigenfunction extrapolation and the link between class labels and ideal kernel eigenfunctions; second, we extend this scheme to the multiple kernel setting to improve the modeling power of the proposed method; third, we compare our approach with state-of-the-art semi-supervised learning methods under both the single kernel and multiple kernel settings and demonstrate the superiority of our approach in terms of both speed and accuracy; fourth, we apply our approach in the indoor WiFi localization problem which is shown to beat the currently best results in the same data set.

The rest of the paper is organized as follows. Section 2 introduces the background and related work. Section 3 presents the proposed label-aware base kernel design method. Section 4 compares the performance of our approach with a number of algorithms based on spectral transformations. And the last section concludes the paper.

2. Background and related work

Given an $n \times n$ kernel matrix K, the graph Laplacian is computed as $\mathcal{L} = D - K$, where $D \in \mathbf{R}^{n\times n}$ is a (diagonal) degree matrix such that $D_{ii} = \sum_{j=1}^{n} K_{ij}$. The normalized graph Laplacian is defined as $\tilde{\mathcal{L}} = \mathbf{I} - D^{-1/2}KD^{-1/2}$, where $\mathbf{I}$ is the identity matrix. The (normalized) graph Laplacian matrix imposes important smoothness constraints over the graph, which has been widely used in spectral clustering [15], image segmentation [16], and feature selection [17]. In particular, its smaller eigenvalues correspond to smoother eigenvectors over the graph, i.e., the entries of the eigenvector corresponding to neighboring samples are close to each other. Such smoothness is very useful for predicting the actual class labels. Based on this property, a general principle is applied in spectral transformation to build semi-supervised kernel [18]:

$$\tilde{K} = \sum_{i=1}^{n} r(\lambda_i)\phi_i\phi_i^\top.$$

Here, λ_i's $(i = 1, 2, \dots, n)$ are eigenvalues of the (normalized) graph Laplacian $\mathcal{L} \in \mathbf{R}^{n\times n}$ sorted in an ascending order, ϕ_i's are the corresponding eigenvectors, and $r(\cdot)$ is a non-increasing function which enforces larger penalty for less smooth eigenvectors. Various choice of the transform $r(\cdot)$ has been proposed in the literature. For example, the diffusion kernel [19] corresponds to $r(\lambda) = \exp(-(\sigma^2/2)\lambda)$; the cluster kernel [20], the eigenvectors ϕ_i's are based on the degree-normalized kernel matrix $\mathcal{S} = D^{-1/2}KD^{-1/2}$.

The *empirical kernel alignment* [13] is a promising tool to evaluate the degree of agreement between a kernel and the learning target, via the use of "ideal kernel" $K^*(\mathbf{x}, \mathbf{z}) = y(\mathbf{x})y(\mathbf{z})$, where $y(\mathbf{x})$ is the target concept (such as the class label chosen from $\{\pm 1\}$ or $\{0, 1\}$ [21]). Given a set of l training examples, corresponding label vector $\mathbf{y} \in \mathbf{R}^{l\times 1}$, and kernel matrix $K \in \mathbf{R}^{l\times l}$, the alignment is computed as

$$A_{K,y} = \frac{\langle K, \mathbf{y}\mathbf{y}^\top\rangle}{l\sqrt{\langle K, K\rangle}},$$

where $\langle K_1, K_2\rangle = \sum_{ij} K_1(\mathbf{x}_i, \mathbf{x}_j)K_2(\mathbf{x}_i, \mathbf{x}_j)$ is the inner product between matrices. It has been shown that the alignment between kernels is sharply concentrated, i.e., a good alignment on the training set will indicate a good alignment on the test set [13]. On the other hand, $A_{K,y}$ is favorably associated with the generalization performance of a classifier (such as the Parzen window estimator) [13]. Therefore, maximizing the alignment of the kernel with the ideal one provides a general and effective way for kernel design. In this paper, we adopt a different alignment criterion between two kernels K and K' from [11]

$$\rho(K, K') = \frac{\langle K_c, K'_c\rangle_F}{\|K_c\|_F \|K'_c\|_F},$$

where K_c is the centralized version of K. This criterion provides a novel concentration bound, and shows the existence of good predictors for kernels with high alignment, in both classification and regression tasks.

The concept of ideal kernel and its implications have led to several successful methods for kernel learning. The common theme of these methods is to use eigenvectors of the kernel matrix to span a set of base kernels, and then optimize the weighting in the combined kernel via maximizing its alignment with the target (or ideal kernel). For example, Christianini et al. proposed to compute the weighting of each base kernel proportional to the inner product of the corresponding eigenvector with the target [13]. Sinha et al. presented a framework for computing sparse combination of the base kernels [14]. Cortes et al. showed that the weighting in the maximal alignment kernel can be solved via quadratic programming [11]. In [7], a semi-definite programming formulation was adopted to learn a kernel matrix $\tilde{K}$ that is maximally aligned with the ideal kernel:

$$\max_{\tilde{K}} \quad \langle \tilde{K}_{tr}, \mathbf{y}\mathbf{y}^\top\rangle,$$
$$\text{subject to} \; \|\tilde{K}\|_F = 1, \; \tilde{K} \succcurlyeq 0 \quad \operatorname{trace}(\tilde{K}) = 1.$$

Here, $\tilde{K} \in \mathbf{R}^{n\times n}$, $\tilde{K}_{tr} \in \mathbf{R}^{m\times m}$ is the sub-block of $\tilde{K}$ corresponding to m labeled samples, and $\mathbf{y} \in \mathbf{R}^{m\times 1}$ is the vector of training labels. In particular, if the kernel matrix $\tilde{K}$ is spanned by the eigenvectors of the Laplacian, i.e., $\tilde{K} = \sum_{i=1}^{n} \mu_i\phi_i\phi_i^\top$, then the formulation will be reduced to a quadratically constrained quadratic programming (QCQP) [7,10]. This avoids the need to choose the parameters in $r(\cdot)$, and the nonparametric transform will make the kernel design more flexible. In [10], an order constraint on the transformed eigenvalue is further considered. which leads to the following optimization problem:

$$\max_{\mu} \quad vec(\mathbf{y}\mathbf{y}^\top)^\top M\mu$$
$$\text{subject to} \|M\mu\| \le 1, \; \mu_i \ge \mu_{i+1} \ge 0, \quad i = 1, 2, \dots, n-1.$$

Here $\mu = [\mu_1\mu_2\dots\mu_n]^\top$, μ_i's are the eigenvalues to be learned, $vec(\cdot)$ is the column vectorization operator of a matrix, $M = [vec(K_{1,tr}), \dots, vec(K_{n,tr})]$ where $K_{i,tr}$ is the sub-block of $K_i = \phi_i\phi_i^\top$ corresponding to the labeled samples, and ϕ_i's are the eigenvectors of the graph Laplacian sorted in ascending order based on the eigenvalues. The order constraint reflects important prior belief that smoother eigenvectors should be given higher priority in building the kernel, and empirically it has shown improved behavior over parametric and purely nonparametric spectral transforms [10]. Recently Cortes et al. [12] proposed the algorithm based on the concept of local Rademacher complexity, which is upper-bounded by tailsum of the eigenvalues of kernels. The authors proposed a regularization formulation for controlling the tailsum instead of the traditional way on restricting trace norm of kernels. Note that the base kernels are not necessarily orthogonal eigenvectors computed from one empirical kernel matrix (or its Laplacian matrix). In many cases the base kernels themselves can be different empirical kernel matrices that are either from different domains/views of the data, or simply computed by varying the kernel parameters, such as [7,11,12]. Most of the algorithms we have reviewed here apply to both cases.

The biggest difference between our approach and existing SSL kernel design methods is that our approach utilizes the given labels to compute a set of more "accurate" eigenvectors to span the base kernels. On the other hand, there are many SSL algorithms whose focus is not on kernel design but instead the estimation of the class labels directly. For example, the local and global consistency method [22] iteratively propagates the labels of X_l to the whole data set by

$$F(t+1) = \alpha S F(t) + (1-\alpha) Y,$$

where F is the class label to be estimated, S is the degree normalized kernel matrix $S = D^{-1/2} K D^{-1/2}$, and Y is the class label (entries of Y, corresponding to unlabeled data, are filled with 0's). This method requires iterative propagation, which is equivalent to performing a matrix inverse; in comparison, we only need one step in extrapolating the ideal kernel eigenvectors. In addition, the local and global consistency method is motivated by explicit "propagation" of labels on a connected graph; in comparison, we base our method on theories of integral operators and eigenfunction extrapolation [23]. In [24], the authors utilized the harmonic property

$$f = D^{-1} K f,$$

where f is the estimated label, K and D are the kernel matrix and the degree matrix, respectively. It states that the label of one sample should be consistent with a linear combination of the labels from its nearby samples. This is very closely related to Eq. (5) (see Section 3.3). However, Zhu et al. used this property as a global constraint, and computed the class labels by solving a linear system. In comparison, in our approach, Eq. (5) can be deemed as utilizing this property only on labeled samples as a way to extrapolate the ideal kernel eigenvector to the whole data set. Considering this interesting connection, we will empirically compare our approach with the local and global consistency method in one task of wireless sensor localization.

3. Methodology

Kernel target alignment is an important criterion widely used in semi-supervised kernel design [13]. A higher alignment with the ideal kernel indicates the existence of a good classifier with a higher probability [13,11]. It can be easily observed that the overall alignment of the mixed kernel depends directly on the individual base kernel alignment scores. For example, it has been shown that the optimized alignment between a kernel $\tilde{K} = \sum_{i=1}^{k} \beta_i v_i v_i^\top$ and the ideal kernel $yy^\top$ is $\tilde{A}(y) = \frac{1}{k}\sqrt{\sum_{i=1}^{k} \langle v_i, y \rangle_F^4}$, where $\langle v_i, y \rangle$ is the alignment for the ith eigenvector. However, in practice, the base kernels spanned by the eigenvectors of the kernel matrix might deviate a lot from the target due to various practical factors, such as noise, choice of kernel types/parameters, or the difficulty of the classification problem.

In the following, we consider building more "accurate" eigenvectors to span better base kernels. Note that one reason of the low quality of the base kernels spanned by kernel eigenvectors is that they are computed regardless of the label. Therefore, we may not expect that the kernel eigen-structures faithfully reflect the target variable. To alleviate this problem, we propose to compute a set of desired "eigenvectors" via extrapolation of the ideal kernel eigenfunction. We first discuss the connection between kernel eigenvectors and class labels, and then introduce the concept of kernel eigenfunction extrapolation to build label-aware kernel eigenvectors.

3.1. Kernel eigenvectors and class labels

Proposition 1. *Given l labeled examples ordered from c classes, with ideal kernel in the form of*

$$K^* = \begin{bmatrix} 11'_{l_1} & 0 & \dots & 0 \\ 0 & 11'_{l_2} & \cdots & 0 \\ \vdots & \dots & \ddots & \vdots \\ 0 & \dots & 0 & 11'_{l_c} \end{bmatrix} \tag{1}$$

where l_i is the size of the ith class. Let $Y \in \mathbf{R}^{l \times c}$ be the class label, i.e., $Y_{ij} = 1$ if $\mathbf{x}_i$ is in class j; and $Y_{ij} = 0$ otherwise. Then the i th non-zero eigenvector of K^* is $(1/\sqrt{l_i}) Y_i$, where Y_i is the ith column of Y.

Proof. Let the eigenvalue decomposition of K^* be $K^* \mathbf{v}^* = \lambda^* \mathbf{v}^*$. Since K^* only has c different rows (orthogonal to each other), it has rank c with $n-c$ zero eigenvalues. Note that the ith entry of $\mathbf{v}^*$ equals $(1/\lambda^*) K^*(i,:) \mathbf{v}^*$, and K^* has a block-wise constant structure. Therefore $\mathbf{v}^*$ is piecewise constant. Write $\mathbf{v}^*$ as

$$[\underbrace{v_1, \dots, v_1}_{l_1} \underbrace{v_2, \dots, v_2}_{l_2}, \dots, \underbrace{v_c, \dots, v_c}_{l_c}]'.$$

Then the eigensystem can be written as an equation group $m_k v_k = \lambda^* v_k$ for $k = 1, 2, \dots, c$. Each equation in it leads to two conditions: $\lambda^* = l_k$, or $v_k = 0$. However, it is impossible to set $\lambda^* = l_k$ for $k = 1, 2, \dots, C$, since the size of different classes can be different. The only feasible way is to set λ^* equal to one of the ml_k's, i.e., $\lambda^* = l_{k_0}$, and at the same time set $v_k = 0$ for all the $k \neq k_0$. There are c different ways to choose k_0, i.e., $k_0 = 1, 2, \dots, c$. For each choice of k_0, the eigenvalue is $\lambda^* = l_{k_0}$; as to the eigenvector, all its entries corresponding to class k $(k \neq k_0)$ will be zero, and the entries corresponding to class k_0 will be $1/\sqrt{l_{k_0}}$ (since they are equal and should normalize to 1). This completes the proof.□

Proposition 1 shows that non-zero eigenvectors of the ideal kernel correspond exactly to the classes labels (up to a scaling). For example, [16] shows that the eigenvectors corresponding to the second smallest eigenvalue of the normalized graph Laplacian provide a relaxed solution for a two-class clustering problem.[1] Therefore, eigenvectors and class labels have intrinsic connections. The main difference is that eigenvectors of the kernel matrix can be noisy and may fail to reveal underlying cluster structures due to their unsupervised nature; in comparison, class label represents prior knowledge and is always a clean, piecewise constant vector. The connection indicates that if we can expand "ideal" kernel eigenvectors from labeled samples to the whole data set and obtain a set of high-quality eigenvectors that align better to class labels, then the resultant base kernels will also have a higher target alignment. To achieve this goal, we need notions of eigenfunction and its extrapolation via the Nyström extension.

[1] Positive entries in this eigenvector will be deemed as positive class and negative entries will be indicative of the negative class.

3.2. Eigenfunction expansion

Let $\mathcal{A}$ be a linear operator on a function space. The eigenfunction f of $\mathcal{A}$ is any non-zero function that returns itself from the operator, i.e., $\mathcal{A}f = \lambda f$, where λ is the eigenvalue. In this paper, we are interested in the case where $\mathcal{A}$ is a symmetric, positive semi-definite kernel $K(\mathbf{x},\mathbf{z})$. The corresponding eigenfunction $\phi(\cdot)$, given the underlying sample distribution $p(\mathbf{x})$, is defined as [25]

$$\int K(\mathbf{x},\mathbf{z})\phi(\mathbf{x})p(\mathbf{x})\,d\mathbf{x} = \lambda\phi(\mathbf{z}). \tag{2}$$

The standard numerical method to approximate the eigenfunctions and eigenvalues in Eq. (2) is to replace the integral with the empirical average [26,25]

$$\int K(\mathbf{x},\mathbf{z})p(\mathbf{x})\phi(\mathbf{x})\,d\mathbf{x} \approx \frac{1}{q}\sum_{i=1}^{q} K(\mathbf{x}_i,\mathbf{z})\phi(\mathbf{x}_i), \tag{3}$$

where $\mathbf{x}_{i, i=1,2,\dots,q}$ is drawn from the distribution $f(\cdot)$. By choosing $\mathbf{z}$ as $\mathbf{z}=\mathbf{x}_i$, $i=1,2,\dots,q$, Eq. (3) extends to a matrix eigenvalue decomposition $K\mathbf{v}=\lambda\mathbf{v}$, where K is the kernel matrix defined as $K_{ij}=K(\mathbf{x}_i,\mathbf{x}_j)$ for $1 \le i,\ j \le q$, and $\mathbf{v}$ is the discrete counterpart of ϕ in that $\phi(\mathbf{x}_i)\approx\mathbf{v}(i)$. Then the eigenfunction can be extended by

$$\phi(\mathbf{z}) \approx \frac{1}{q\lambda}\sum_{i=1}^{q} K(\mathbf{z},\mathbf{x}_i)\mathbf{v}(i). \tag{4}$$

This is known as the Nyström extension [23], which means that the eigenvectors of the empirical kernel matrix evaluated on a finite sample set can be used as approximators to the whole eigenfunction of the linear operator. Interestingly, Eq. (4) is proportional to the projection of a test point computed in kernel PCA [27]. The approximation can be justified by examining the convergence of eigenvalues and eigenvectors as the number of examples increases [28,27].

3.3. Extrapolating ideal kernel eigenfunctions

Motivated by the eigenfunction extension, we propose to extrapolate the ideal kernel eigenvectors as follows. Suppose we are given the labeled set $X_l=\{\mathbf{x}_i\}_{i=1}^{l}$ with labels $Y\in\mathbf{R}^{l\times c}$, where c is the number of classes, and the unlabeled set $X_u=\{\mathbf{x}_i\}_{i=l+1}^{n}$. Then, in order to expand the ideal kernel eigenfunction from X_l to the whole data set $X_l\cup X_u$, we can choose $\{\mathbf{x}_i\}_{i=1}^{q}$ in (4) as X_l, choose $\mathbf{z}$ in (4) as $X_l\cup X_u$, and choose $\mathbf{v}(i)$ as the labels of X_l. Suppose the estimated kernel eigenvectors are denoted as $u_k\in\mathbf{R}^{n\times 1} for k=1,2,\dots,c$, corresponding to the c classes, then we have

$$u_k(i) = \frac{1}{l\lambda_k}\sum_{\mathbf{x}_j\in X_l} K(\mathbf{x}_i,\mathbf{x}_j)Y_{jk}. \tag{5}$$

Here, λ_k is the eigenvalue corresponding to the kth class, which according to Proposition 1 is proportional to the size of the kth class. To guarantee that the estimated labels/eigenvector entries are in a reasonable range, one can also normalize the weighting coefficients $K(\mathbf{x}_i,\mathbf{x}_j)$ by $\sum_j K(\mathbf{x}_i,\mathbf{x}_j)$. The advantage of extrapolating the ideal kernel eigenfunction is that the resultant eigenvector incorporates label information directly. Therefore, empirically they typically have higher alignment with the target compared with the eigenvectors of the kernel matrix, the computation of the latter being totally irrespective of available class labels. With such label-aware eigenvectors, we will then have better base kernels for semi-supervised learning.

3.4. Combining base kernels

Having obtained a set of extrapolated ideal kernel eigenvectors, we can use them to span base kernels for semi-supervised kernel design. In case the number of labeled sample is very limited, using label-aware eigenvectors alone may not be sufficient. Therefore, it is safer to incorporate the kernel eigenvectors as well. Suppose we have obtained a set of c extrapolated eigenvectors $u_1,u_2,\dots,u_c$, as well as a set of k eigenvectors $v_1,v_2,\dots,v_k$, from the kernel matrix (or graph Laplacian). Then we want to learn the following kernel:

$$\tilde{K} = \sum_{i=1}^{c}\alpha_i u_i u_i^{\top} + \sum_{j=1}^{k}\beta_j v_j v_j^{\top}. \tag{6}$$

The mixing coefficients can be determined by maximizing the alignment to the target. In other words, it will be automatically determined which parts take higher weights. If the problem is easy and kernel eigenvectors already are accurate enough, then they will play a major role in shaping the new kernel; on the other hand, if the kernel eigenvectors turn out to be noisy and poorly aligned to the target, then the label-aware eigenvectors will probably assume higher weights. In the literature, there are various ways to compute the weights such as uniform weighting, independent alignment-based weighting, or the quadratic programming approach. In this paper, we adopt a well-known method *alignf* [11] and a very recent method *local rademacher complexity* [12] that determines the mixture weights jointly by seeking to maximize the alignment between the convex combination kernel and the target kernel.

With the learned kernel $\tilde{K}$, one can use $\tilde{K}$ as the similarity matrix and plug it in Support Vector Machine (SVM) for training and testing.

The whole algorithm is summarized in Algorithm 1.

Algorithm 1. Input: labeled samples $X_l=\{\mathbf{x}_i\}_{i=1}^{l}$, unlabeled sample set $X_u=\{\mathbf{x}_i\}_{i=l+1}^{n}$; Gaussian Kernel $k(\cdot,\cdot)$, label $Y=[\mathbf{y}_1,\mathbf{y}_2,\dots,\mathbf{y}_c]\in\mathbf{R}^{l\times c}$.

```
1. Compute the kernel matrix defined among $X_l \cup X_u$ and $X_l$, as
   $K_{nl}\in\mathbf{R}^{n\times l}$; compute the degree matrix $D_n = diag(K_{nl}\cdot 1_{l\times 1})$;
2. Perform eigenfunction extrapolation as
   $[u_1,u_2,\dots,u_c] = D_n^{-1}K_{nl}Y$;
3. Use the Nyström method [26] to compute eigenvectors
   corresponding to dominant $k$ eigenvalues of kernel matrix
   or diminishing $k$ eigenvalues of the (normalized) graph
   Laplacian, as $[v_1,v_2,\dots,v_k]$;
4. Compute the weights of the base eigenvectors
   $[u_1,u_2,\dots,u_c,v_1,v_2,\dots,v_k]$;
5. Compute the new kernel $\tilde{K} = \sum_{i=1}^{c}\alpha_i u_i u_i^{\top} + \sum_{j=1}^{k}\beta_j v_j v_j^{\top}$;
6. Apply kernel $\tilde{K}$ for training and testing.
```

3.5. Multiple kernel setting

In the previous section, we only consider the use of a single (empirical) kernel matrix K to construct base kernels in semi-supervised kernel learning. Recently, researchers have emphasized the need to consider multiple kernels that may correspond to heterogeneous data sources (or views) and can improve the model flexibility [11,12,7]. In case no physically meaningful multiple domains exist, one can always artificially create them. For example, by changing the kernel width parameter, multiple RBF kernel matrices can be constructed [11,12,7]. Then these different empirical kernel matrices can all be used to construct base kernels (or themselves can be directly used as base kernels) for ultimate kernel learning.

In this section, we incorporate this idea in our method. Suppose we have a number of p different kernel matrices (or graph Laplacians), corresponding to p different sources. Then, we will compute both the unsupervised kernel eigenvectors (u_i's, $i=1,2,\ldots,c$) and the label-aware kernel eigenvectors (v_j's, $j=1,2,\ldots,k$) for each kernel. Ultimately, all these eigenvectors are fed together into a multiple kernel learning procedure. More specifically, we can write the final kernel as

$$\tilde{K} = \sum_{t=1}^{p}\left(\sum_{i=1}^{c}\alpha_{ti}u_{ti}u_{ti}^{\top} + \sum_{j=1}^{k}\beta_{tj}v_{tj}v_{tj}^{\top}\right) \tag{7}$$

Next, altogether $p(k+c)$ base kernels can be fed into a kernel learning procedure such as *alignf* procedure [11] or *local rademacher complexity* [12] to determine the mixing weights.

3.6. Complexity

In Algorithm 1, steps 1 and 2 take $O(nc)$ time and space, where n is the sample size and c the number of classes; step 3 takes $O(np^2)$ time and $O(np)$ space; step 4 takes $O(lc)$ time and space; in applying the learned kernel $\tilde{K}$ in SVM, we only need the $l \times l$ block of $\tilde{K}$ corresponding to labeled samples, and the $u \times l$ block corresponding to the block between unlabeled and labeled samples. Therefore, the space needed is $O(nl)$. Step 5 takes $O(cl)$ time. In step 6, the training takes empirically $O(l^{2.3})$ time using the libsvm package, and testing takes $O(pn+ln)$, and the time complexity is $O(nl+np^2+l^{2.3})$. In practice, we have $l, p \ll n$. Therefore, overall our algorithm has a linear time and space complexities.

4. Experiments

This section compares our method with a number of state-of-the-art kernel design algorithms for semi-supervised learning tasks including classification (Section 4.1) and regression (Section 4.2). Furthermore, we examine alignment scores of base kernels obtained via different methods, and compare the classification performance of traditional unsupervised base kernels with our method side by side in the settings of single kernel and multiple kernels (Section 4.3). Our codes are written in Matlab and run on a Intel (R) Core (TM) i5 CPU @2.60 GHZ 2.60 GHZ PC with 8 GB RAM.

4.1. Comparison with other semi-supervised kernel learning methods in classification

In this section, we compare the following semi-supervised kernel design methods: (1) cluster kernel [20], where $r(\cdot)$ is chosen as linear function $r(\lambda)=\lambda$; (2) diffusion kernel $r(\lambda)=\exp(-\lambda/\delta)$ [19]; (3) maximal alignment kernel [13] using the top $0.1n$ eigenvectors from the kernel matrix; (4) our approach; and (5) non-parametric graph kernel [10] using the first $p=0.1n$ eigenvectors from the normalized Laplacian $\tilde{\mathcal{L}}$. Evaluation is based on the alignment on the unlabeled data, and classification error of SVM using the learned kernel. Here, for a fair comparison, our method only uses one empirical kernel matrix instead of multiple empirical kernel matrix altogether.

We used the Gaussian kernel $K(\mathbf{x}_1,\mathbf{x}_2)=\exp(-\|\mathbf{x}_1-\mathbf{x}_2\|^2 \cdot b)$ in all our experiments. In semi-supervised learning parameter selection is an open problem. In this work, the parameters are chosen as follows. For the kernel width, we first compute b_0 as the inverse of the average squared pairwise distances, and then choose b among $b_0 \cdot \{\frac{1}{50},\frac{1}{25},\frac{1}{10},\frac{1}{5},1,5,10\}$ that gives the best performance. The parameters δ and ϵ are chosen from $\{10^{-5},10^{-3},10^{-1},1\}$. Each algorithm is repeated 30 times with 50 labeled samples randomly chosen for each class. Cluster kernel method and non-parametric graph kernel method use $10\%n$ diminishing eigenvectors from the normalized graph Laplacian; other methods use the top 10% eigenvectors of the kernel matrix. Results are reported in Table 1. As can be seen, our algorithm gives competitive performance and at the same time very efficient.

4.2. Comparison with other semi-supervised kernel learning methods in regression

In this section, we report empirical results of our algorithm in kernel based regression problems. The task is indoor location

Table 1
Classification performance using different semi-supervised kernel design schemes. For each cell, the top row is the mean/std of the kernel alignment score (in [0,1]) on the test set, and in bracket is the averaged time consumption (in seconds); the bottom row is the mean/std of classification error (%).

Data size/dim	Spectral graph kernel	Ours	Cluster kernel linear	Diffusion kernel	Max-alignment kernel
Digit1	0.29 ± 0.07 (84.9)	0.82 ± 0.02 (1.2)	0.13 ± 0.005 (2.4)	0.10 ± 0.001 (13.0)	0.14 ± 0.001 (12.6)
1500×241	4.31 ± 1.93	4.89 ± 0.85	5.37 ± 1.23	6.13 ± 1.63	$\mathbf{3.82} \pm \mathbf{1.23}$
USPS	0.23 ± 0.08 (74.9)	0.66 ± 0.04 (1.2)	0.43 ± 0.001 (2.5)	0.06 ± 0.001 (16.0)	0.06 ± 0.01 (12.7)
1500×241	7.47 ± 4.41	$\mathbf{6.64} \pm \mathbf{1.27}$	$\mathbf{6.56} \pm \mathbf{1.02}$	7.27 ± 0.59	9.81 ± 0.49
COIL2	0.11 ± 0.005 (73.4)	0.55 ± 0.07 (1.2)	0.10 ± 0.001 (2.4)	0.05 ± 0.003 (8.4)	0.07 ± 0.00 (5.3)
1500×241	18.49 ± 2.47	$\mathbf{13.44} \pm \mathbf{2.41}$	18.51 ± 4.66	19.08 ± 2.05	19.32 ± 1.89
BCI	0.07 ± 0.003 (9.9)	0.14 ± 0.04 (0.4)	0.04 ± 0.001 (0.2)	0.07 ± 0.003 (0.4)	0.07 ± 0.002 (0.5)
400×241	$\mathbf{32.95} \pm \mathbf{3.38}$	$\mathbf{32.99} \pm \mathbf{3.10}$	42.02 ± 2.89	33.58 ± 2.83	34.85 ± 2.75
COIL	0.01 ± 0.001 (199.5)	0.11 ± 0.05 (0.4)	0.08 ± 0.002 (2.58)	0.06 ± 0.001 (8.3)	0.07 ± 0.001 (5.5)
1500×241	21.90 ± 3.24	$\mathbf{9.14} \pm \mathbf{0.96}$	10.89 ± 1.12	11.67 ± 1.43	11.75 ± 1.49
g241n	0.40 ± 0.003 (108.2)	0.33 ± 0.03 (1.4)	0.03 ± 0.007 (2.5)	0.04 ± 0.00 (20.3)	0.04 ± 0.00 (6.7)
1500×241	$\mathbf{13.64} \pm \mathbf{1.28}$	24.11 ± 1.73	26.59 ± 3.96	19.68 ± 1.52	18.61 ± 1.75
Text	0.13 ± 0.01 (181.0)	0.30 ± 0.02 (20.1)	0.03 ± 0.001 (68.1)	0.03 ± 0.00 (208.0)	0.03 ± 0.004 (130.7)
1500×11960	25.55 ± 1.65	$\mathbf{23.42} \pm \mathbf{1.46}$	32.90 ± 6.64	24.89 ± 1.81	26.78 ± 4.88
usps38	0.48 ± 0.004 (77.3)	0.84 ± 0.02 (1.2)	0.12 ± 0.001 (1.6)	0.11 ± 0.001 (6.8)	0.11 ± 0.001 (4.5)
1200×256	4.82 ± 1.33	$\mathbf{2.82} \pm \mathbf{0.83}$	5.10 ± 0.89	6.06 ± 1.01	6.06 ± 0.85
usps49	0.40 ± 0.13 (82.1)	0.86 ± 0.01 (1.2)	0.09 ± 0.001 (1.9)	0.08 ± 0.001 (9.3)	0.07 ± 0.001 (8.9)
1296×256	2.83 ± 0.92	$\mathbf{1.98} \pm \mathbf{0.52}$	6.29 ± 2.11	8.26 ± 0.83	10.67 ± 1.24
usps56	0.48 ± 0.06 (80.0)	0.86 ± 0.01 (1.2)	0.12 ± 0.001 (1.7)	0.09 ± 0.003 (18.2)	0.11 ± 0.001 (5.0)
1220×256	2.87 ± 0.92	$\mathbf{2.44} \pm \mathbf{0.59}$	3.89 ± 1.57	3.85 ± 0.97	5.79 ± 1.06
usps27	0.58 ± 0.004 (101.8)	0.91 ± 0.06 (1.2)	0.37 ± 0.001 (2.3)	0.10 ± 0.001 (11.8)	0.13 ± 0.001 (6.9)
1376×256	1.79 ± 0.42	$\mathbf{1.21} \pm \mathbf{0.25}$	1.80 ± 0.25	2.28 ± 0.56	4.80 ± 1.29
odd/even	0.21 ± 0.008 (419.0)	0.65 ± 0.03 (1.6)	0.12 ± 0.001 (8.8)	0.03 ± 0.004 (38.5)	0.08 ± 0.00 (22.3)
2007×256	10.14 ± 2.11	$\mathbf{9.58} \pm \mathbf{1.56}$	14.59 ± 1.49	14.08 ± 2.04	15.64 ± 2.91

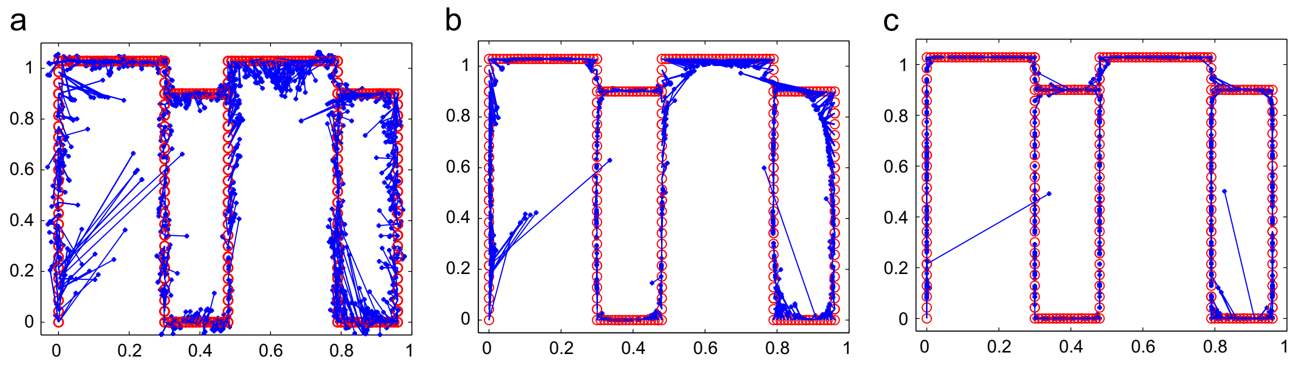


Fig. 1. Localization results by different methods. For each test point, a line is connected between the true and the estimated location/coordinates. (a) standard SVR. (b) Gaussian Fields. (c) our method. (For interpretation of the references to color in the text, the reader is referred to the web version of this paper.)

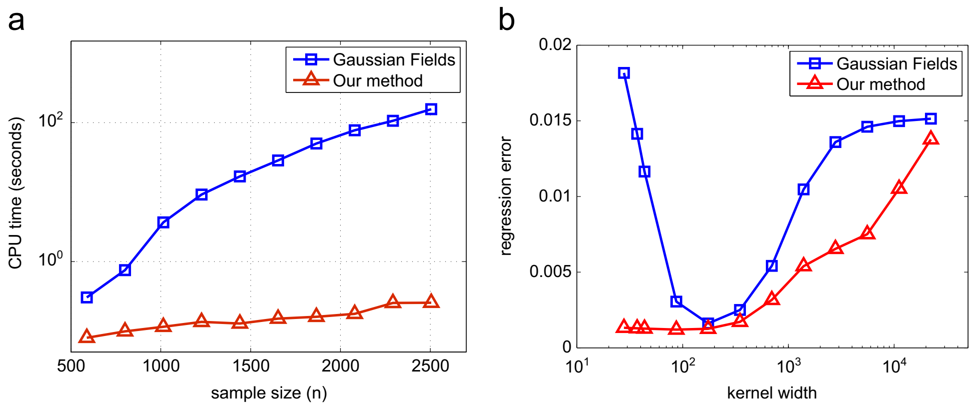


Fig. 2. Properties of the Gaussian Fields based method and our approach. (a) Time versus sample size. (b) Error versus kernel width.

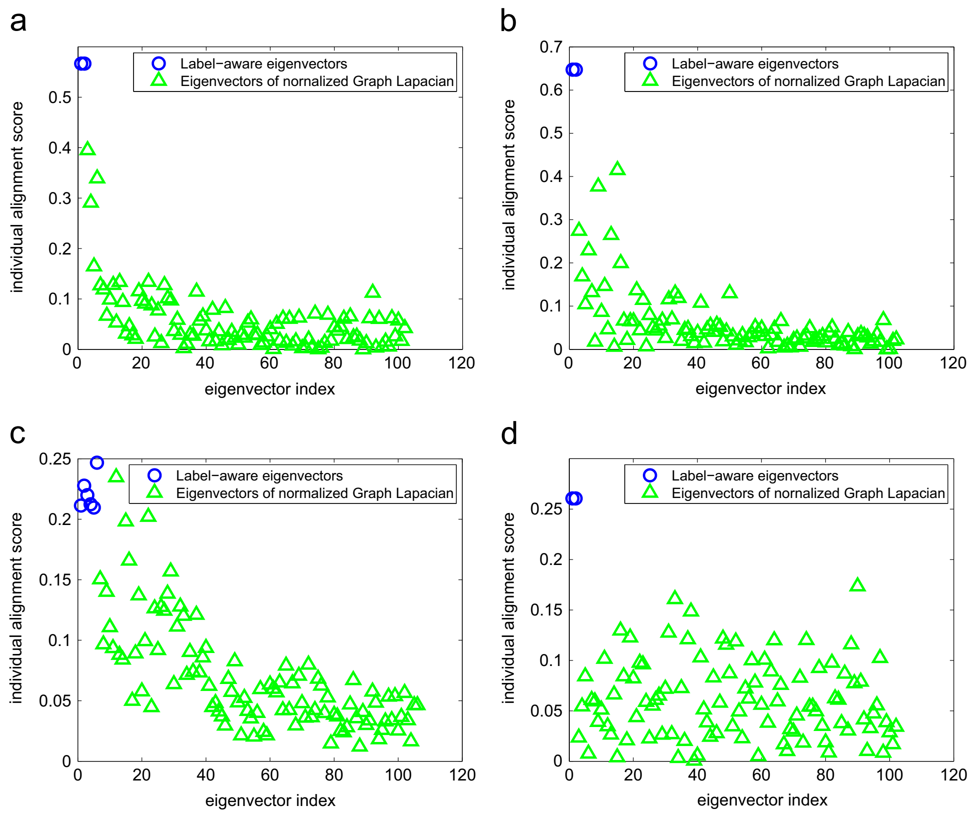


Fig. 3. The individual target alignment score of label-aware base eigenvectors and the traditional kernel eigenvectors *on the unlabeled data*. For simplicity of visualization, here the reported score is the average alignment between one eigenvector and all the *c* target variables/classes. (a) Text (2 classes). (b) USPS (2 classes). (c) Coil (6 classes). (d) BCI (2 classes). (For interpretation of the references to color in this figure caption, the reader is referred to the web version of this paper.)

Table 2
Side-by-side comparison of the learning performance using single empirical kernel matrix.

Data size/dim	alignf (baseline)	alignf (ours)	local rademacher complexity (baseline)	local rademacher complexity (ours)
Digit1	0.27 ± 0.02 (0.8)	0.82 ± 0.02 (1.2)	0.49 ± 0.02(16.9)	0.82 ± 0.02(17.1)
1500×241	10.73 ± 0.54	4.89 ± 0.85	6.45 ± 0.63	5.18 ± 0.56
USPS	0.02 ± 0.01 (0.5)	0.66 ± 0.04 (1.2)	0.03 ± 0.01 (6.5)	0.72 ± 0.01(5.7)
1500×241	14.15 ± 1.92	6.64 ± 1.27	10.09 ± 1.49	8.18 ± 1.26
COIL2	0.15 ± 0.07 (0.6)	0.55 ± 0.07 (1.2)	0.18 ± 0.01(13.2)	0.51 ± 0.03 (11.3)
1500×241	19.92 ± 2.14	13.44 ± 2.41	18.54 ± 3.43	14.22 ± 2.12
BCI	0.01 ± 0.04 (0.6)	0.14 ± 0.04 (0.4)	0.06 ± 0.01(17.2)	0.18 ± 0.04(14.3)
400×241	51.36 ± 3.50	32.99 ± 3.10	39.36 ± 3.35	33.09 ± 2.59
COIL	0.09 ± 0.01 (0.3)	0.11 ± 0.05 (0.4)	0.08 ± 0.01 (20.3)	0.12 ± 0.05 (20.4)
1500×241	15.92 ± 2.31	9.14 ± 0.96	19.94 ± 4.32	10.65 ± 4.33
g241n	0.31 ± 0.01 (0.6)	0.33 ± 0.03 (1.4)	0.20 ± 0.02(14.5)	0.33 ± 0.01(17.8)
1500×241	29.03 ± 0.67	24.11 ± 1.73	26.35 ± 1.16	23.90 ± 1.16
Text	0.13 ± 0.02 (18.8)	0.30 ± 0.02 (20.1)	0.14 ± 0.03(25.6)	0.34 ± 0.02(22.1)
1500×11960	30.18 ± 2.28	23.42 ± 1.46	27.34 ± 2.87	23.09 ± 2.21
usps38	0.53 ± 0.01 (0.9)	0.84 ± 0.02 (1.2)	0.56 ± 0.02(25.6)	0.85 ± 0.03(23.6)
1200×256	6.25 ± 0.80	2.82 ± 0.83	4.63 ± 0.41	2.54 ± 0.32
usps49	0.52 ± 0.02 (0.7)	0.86 ± 0.01 (1.2)	0.37 ± 0.01(25.0)	0.87 ± 0.02(20.9)
1296×256	5.70 ± 0.77	1.98 ± 0.52	4.63 ± 0.24	2.09 ± 0.39
usps56	0.52 ± 0.02(0.7)	0.86 ± 0.01 (1.2)	0.55 ± 0.03(39.4)	0.86 ± 0.05(22.5)
1220×256	4.09 ± 0.72	2.44 ± 0.59	5.54 ± 0.52	2.36 ± 0.53
usps27	0.56 ± 0.01 (0.8)	0.91 ± 0.06 (1.2)	0.78 ± 0.02(5.1)	0.88 ± 0.01(19.1)
1376×256	1.98 ± 0.16	1.21 ± 0.25	2.09 ± 0.31	1.18 ± 0.27
odd/even	0.22 ± 0.01 (0.8)	0.65 ± 0.03 (1.6)	0.31 ± 0.02(27.0)	0.70 ± 0.02(14.1)
2007×256	13.91 ± 1.52	9.58 ± 1.56	10.27 ± 0.64	8.82 ± 0.84

Table 3
Side-by-side comparison of the learning performance using multiple empirical kernel matrix.

Data size/dim	alignf (baseline)	alignf (ours)	local rademacher complexity (baseline)	local rademacher complexity (ours)
Digit1	0.82 ± 0.02 (1.2)	0.86 ± 0.02(2.8)	0.82 ± 0.02(17.1)	0.79 ± 0.01(30.8)
1500×241	4.89 ± 0.85	4.70 ± 0.92	5.18 ± 0.56	4.54 ± 0.61
USPS	0.66 ± 0.04 (1.2)	0.72 ± 0.04(2.7)	0.72 ± 0.01(5.7)	0.73 ± 0.01(15.1)
1500×241	6.64 ± 1.27	6.09 ± 1.02	8.18 ± 1.26	7.04 ± 1.04
COIL2	0.55 ± 0.07 (1.2)	0.58 ± 0.02 (3.2)	0.51 ± 0.03 (11.3)	0.53 ± 0.04 (34.1)
1500×241	13.44 ± 2.41	13.14 ± 2.66	14.22 ± 2.12	14.02 ± 2.66
BCI	0.14 ± 0.04 (0.4)	0.22 ± 0.04(1.4)	0.18 ± 0.04(14.3)	0.22 ± 0.04(25.7)
400×241	32.99 ± 3.10	30.27 ± 2.02	33.09 ± 2.59	30.45 ± 1.88
COIL	0.11 ± 0.05 (0.4)	0.13 ± 0.01 (2.1)	0.12 ± 0.05 (20.4)	0.15 ± 0.03 (42.3)
1500×241	9.14 ± 0.96	9.04 ± 0.98	10.65 ± 4.33	10.61 ± 4.81
g241n	0.33 ± 0.03 (1.4)	0.37 ± 0.03(5.6)	0.33 ± 0.01(17.8)	0.34 ± 0.01(24.5)
1500×241	24.11 ± 1.73	22.36 ± 1.93	23.90 ± 1.16	23.09 ± 1.01
Text	0.30 ± 0.02 (20.1)	0.33 ± 0.02(23.8)	0.34 ± 0.02(22.1)	0.34 ± 0.01(46.3)
1500×11960	23.42 ± 1.46	23.36 ± 0.96	23.09 ± 2.21	23.00 ± 1.13
usps38	0.84 ± 0.02 (1.2)	0.81 ± 0.02(2.7)	0.85 ± 0.03(23.6)	0.84 ± 0.02(54.7)
1200×256	2.82 ± 0.83	2.63 ± 0.33	2.54 ± 0.32	2.27 ± 0.45
usps49	0.86 ± 0.01 (1.2)	0.86 ± 0.01(2.9)	0.87 ± 0.02(20.9)	0.87 ± 0.01(23.7)
1296×256	1.98 ± 0.52	1.81 ± 0.22	2.09 ± 0.39	1.90 ± 0.45
usps56	0.86 ± 0.01 (1.2)	0.87 ± 0.01(1.6)	0.86 ± 0.05(22.5)	0.85 ± 0.01(44.7)
1220×256	2.44 ± 0.59	2.35 ± 0.59	2.36 ± 0.53	2.27 ± 0.32
usps27	0.91 ± 0.06 (1.2)	0.90 ± 0.01(1.9)	0.88 ± 0.01(19.1)	0.90 ± 0.01(35.6)
1376×256	1.21 ± 0.25	0.72 ± 0.21	1.18 ± 0.27	1.00 ± 0.19
odd/even	0.65 ± 0.03 (1.6)	0.67 ± 0.02(3.1)	0.60 ± 0.02(14.1)	0.67 ± 0.02(21.9)
2007×256	9.58 ± 1.56	8.90 ± 1.43	10.36 ± 0.84	9.36 ± 0.84

estimation using received signal strength (RSS) that a client device received from Wi-Fi access points [29]. We compare our result with Gaussian Fields based method [24]. In particular, we adopt the support vector regression (SVR) [30] that works on the learned kernel in Algorithm 1. We normalize the labels y_i's such that they scale in the range $[0, 1]$. We use the Gaussian kernel in the experiments. The kernel width is selected in a similar way as the classification tasks. For our method, we set $\epsilon = 0.05$ in the support vector regression setting. The regularization parameter C is chosen as $\{0.1, 1, 10, 100, 1000, 10\,000\}$, similar to settings in [31].

In Fig. 1, we plot the regression results on the 2-D plane. Here, red circles are the true coordinates, and blue dots are estimated ones. A line is connected between every pair of true and estimated points. As can be seen, our approach provides better localization results compared with Gaussian Fields based method. We use the square root of the mean squared error to measure the regression quality The error of standard SVR is 2.5×10^{-3}; that of Gaussian Fields based method is 1.61×10^{-3}; while ours is only around 1.19×10^{-3}. Our regression error is reduced by about 25% compared with Gaussian Fields based method, and more than 50% compared with the standard supervised SVR. In Fig. 2(a), we gradually increase the number of unlabeled samples from 200 to 2000, and examine the time consumption. As can be seen, our approach is orders of magnitudes faster compared with Gaussian Fields based method. In Fig. 2(b), we plot the regression error of the two methods with regard to the Gaussian kernel width. As can be seen, our approach is less sensitive to the choice of the kernel parameters. This makes it a practical in real-world applications. From this example, we can see that semi-supervised kernel design can give competitive performance compared with state-of-the-art SSL algorithms that focus on estimating the labels (but not learning a kernel). This validates the importance of a good kernel in semi-supervised learning tasks. Of course, there are many SSL algorithms whose focus is not on learning kernel. We choose the Gaussian Fields based method as an example for comparison because it has shown to provide state-of-the-art results in this localization task [29].

4.3. *Evaluation of superiority of label-aware base kernels*

In previous sections, we have demonstrated that the proposed method outperforms existing methods in semi-supervised kernel learning tasks including classification and regression. To further verify that this superiority is attributed to the newly proposed label-aware base kernels, in this section, we examine in more detail the usefulness of the proposed label-aware base kernels in two different ways. First, we examine the individual alignment score of the base kernels as an index of their quality. Second, we apply state-of-the-art kernel combination schemes [11,12] on two sets of base kernels, one is the traditional (unsupervised) base kernels, and the other includes the label-aware base kernel, and compare their learning performance. We call this side-by-side comparison. This comparison clearly shows that by adding the newly proposed base kernels, the learning performance of the mixed kernel improves.

4.3.1. *Alignment score comparison*

In Fig. 3, we examine alignment score of the label-aware eigenvectors (blue circles) and those from the normalized Graph Laplacian.[2] Here, the reported score is the average alignment between one eigenvector and all the c target variables. As can be seen, the label-aware eigenvectors almost always have higher or at least very similar alignment scores compared with the eigenvectors of the graph Laplacian.

4.3.2. *Side-by-side learning performance comparison*

In this section, we perform a side-by-side comparison among two sets of base kernels to examine their performance in kernel learning. The first set only contains unsupervised kernel eigenvectors; the second set further adds label-aware base kernels. For both settings, we use Gaussian kernels $K(\mathbf{x}_1, \mathbf{x}_2) = \exp(-\|\mathbf{x}_1 - \mathbf{x}_2\|^2 \cdot b)$ as base kernels. For each base kernel, we extract p eigenvectors from its graph Laplacian, where p is chosen among $0.01n, 0.05n, 0.1n, 0.15n, 0.2n$. Here, algnf [11]and local rademacher [12] are chosen as the baseline kernel combination schemes.

The kernel parameter and regularization parameter C in SVM are the same as Section 4. In the local rademacher complexity method, parameters θ and C are chosen among $\{0.1n, 0.5n, 0.8n, 0.9n, n\}$ and $\{0.1, 1, 10, 100, 1000, 10\,000\}$. We examine the use of a single empirical kernel matrix (see Table 2), as well as a set of multiple empirical kernel matrices whose kernel width parameters vary in $b_0 \cdot \{\frac{1}{50}, \frac{1}{25}, \frac{1}{10}, \frac{1}{5}, 1, 5, 10\}$ (see Table 3). As can be seen, by incorporating the label-aware base kernels, the learning performances are improved in most of the data sets in both kernel combination schemes. We also note that the improvement is less significant in case multiple empirical kernel matrices are used. We speculate that when more empirical kernel matrices are used (with varying kernel parameter), they may have higher chances to align to the target. However, in some data sets, such as usps38, and usps49, although multiple empirical kernel matrices are used which lead to more base kernels, their performances can still be significantly improved after adding the label aware base kernels. We speculate that these data sets are very difficult, and even under a wider choice of the kernel parameters, the resultant unsupervised empirical kernel matrices are still poor candidate for base kernels, and have to rely on using the label information to further improve their quality. This clearly demonstrate the usefulness of the proposed label-aware base kernels in difficult learning tasks.

[2] Empirically, eigenvectors from the normalized graph Laplacian have higher target alignment than those from the kernel matrix.

5. Conclusion

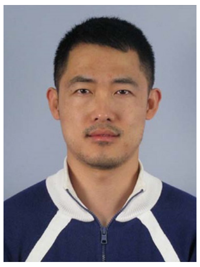
This paper proposed a new algorithm for semi-supervised kernel design. Unlike traditional methods that use kernel eigenvectors to span the base kernel and focus on tuning their weights, this work aims at designing high-quality base kernel. In particular, we compute the label-aware eigenvectors via extending the ideal kernel eigenfunction. The experimental results demonstrated the superiority of our algorithm. An important direction for our future research is to theoretically study the alignment of the label-aware base kernels. In addition, we would explore different ways for propagating the ideal kernel and combining multiple kernels from multiple sources.

References

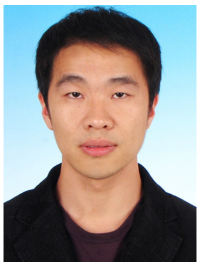
[1] M. Belkin, P. Niyogi, V. Sindhwani, Manifold regularization: a geometric framework for learning from labeled and unlabeled examples, J. Mach. Learn. Res. 7 (2006) 2399–2434.
[2] B. Kulis, S. Basu, I. Dhillon, R. Mooney, Semi-supervised graph clustering: a kernel approach, in: Proceedings of the 22th Annual International Conference on Machine Learning, 2005.
[3] O. Chapelle, A. Zien, Semi-supervised classification by low density separation, in: Proceedings of the 10th International Conference on Artificial Intelligence and Statistics, 2005.
[4] R. Collobert, F. Sinz, J. Weston, L. Bottou, T. Joachims, Large scale transductive svms, J. Mach. Learn. Res. 7 (2006) 1687–1712.
[5] S. Melacci, M. Belkin, Laplacian support vector machines trained in the primal, J. Mach. Learn. Res. 12 (2011) 1149–1184.
[6] T. Joachims, Transductive inference for text classification using support vector machines, in: Proceedings of the 16th Annual International Conference on Machine Learning, 1999.
[7] G.R.G. 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.
[8] Y. Li, Z. Zhou, Towards making unlabeled data never hurt, in: Proceedings of the 28th Annual International Conference on Machine Learning, 2011.
[9] X. Zhou, M. Belkin, Semi-supervised learning by higher order regularization, in: The 14th International Conference on Artificial Intelligence and Statistics, 2011.
[10] X. Zhu, J. Kandola, Z. Ghahramani, J. Lafferty, Nonparametric transforms of graph kernels for semi-supervised learning, in: Advances in Neural Information Processing Systems, 2004.
[11] C. Cortes, M. Mohri, A. Rostamizadeh, Two-stage learning kernel algorithms, in: Proceedings of the 27th Annual International Conference on Machine Learning, 2010.
[12] C. Cortes, M. Kloft, M. Mohri, Learning kernels using local rademacher complexity, in: Advances in Neural Information Processing Systems, 2013.
[13] N. Cristianini, J. Kandola, A. Elisseeff, J. Shawe-Taylor, On kernel-target alignment, in: Advances in Neural Information Processing Systems, 2002.
[14] K. Sinha, M. Belkin, Semi-supervised learning using sparse eigenfunction bases, in: Advances in Neural Information Processing Systems, 2009.
[15] A. Y. Ng, M. I. Jordan, Y. Weiss, On spectral clustering: Analysis and an algorithm, in: Advances in Neural Information Processing Systems, 2001.
[16] J. Shi, J. Malik, Normalized cuts and image segmentation, IEEE Trans. Pattern Anal. Mach. Intell. 22 (2000) 731–737.
[17] X. He, M. Ji, C. Zhang, H. Bao, A variance minimization criterion to feature selection using Laplacian regularization, IEEE Trans. Pattern Anal. Mach. Intell. 33 (2011) 2013–2025.
[18] A.J. Smola, R. Kondor, Kernels and regularization on graphs, in: Conference on Learning Theory, 2003.
[19] R.I. Kondor, J.D. Lafferty, Diffusion kernels on graphs and other discrete input spaces, in: Proceedings of the 19th Annual International Conference on Machine Learning, 2002.
[20] O. Chapelle, J. Weston, B. Scholkopf, Cluster kernels for semi-supervised learning, in: Advances in Neural Information Processing Systems, 2003.

[21] J.T. Kwok, I.W. Tsang, Learning with idealized kernels, in: Proceedings of the 20th Annual International Conference on Machine Learning, 2003.
[22] D. Zhou, O. Bousquet, T.N. Lal, J. Weston, B. Scholkopf, Learning with local and global consistency, in: Advances in Neural Information Processing Systems, 2004.
[23] C. Williams, M. Seeger, Using the Nystrom method to speed up kernel machines, in: Advances in Neural Information Processing Systems, 2001.
[24] X. Zhu, Z. Ghahramani, J. Lafferty, Semi-supervised learning using Gaussian fields and harmonic functions, in: Proceedings of the 20th Annual International Conference on Machine Learning, 2003.
[25] C. Williams, M. Seeger, The effect of the input density distribution on kernel-based classifiers, in: Proceedings of the 17rd Annual International Conference on Machine learning, 2000.
[26] C. Fowlkes, S. Belongie, F. Chung, J. Malik, Spectral grouping using the Nystrom method, IEEE Trans. Pattern Anal. Mach. Intell. 26 (2004) 214–225.
[27] Y. Bengio, O. Delalleau, N.L. Roux, J.-F. Paiement, P. Vincent, M. Ouimet, Learning eigenfunctions links spectral embedding and kernel pca, Neural Comput. 16 (2004) 2197–2219.
[28] J. Shawe-taylor, R. Holloway, C.K.I. Williams, The stability of kernel principal components analysis and its relation to the process eigenspectrum, in: Advances in Neural Information Processing Systems, 2003.
[29] Q. Yang, S.J. Pan, V.W. Zheng, Estimating location using wi-fi, IEEE Intell. Syst. 23 (2008) 8–13.
[30] A.J. Smola, B. Scholkopf, A tutorial on support vector regression, J. Stat. Comput. 14 (2004) 199–222.
[31] K. Zhang, L. Lan, J.T. Kwok, S. Vucetic, B. Parvin, Scaling up graph-based semisupervised learning via prototype vector machines, IEEE Trans. Neural Netw. Learn. Syst. 26 (2015) 444–457.

Qiaojun Wang is a Ph.D. student in the Department of Electrical and Computer Engineering at Rutgers University. He received B.E. degree (2007) from Shandong Polytechnic University, China and M.S. degree (2009) from Stevens Institute of Technology, NJ. His research interests include semi-supervised learning, transfer learning and feature selection.

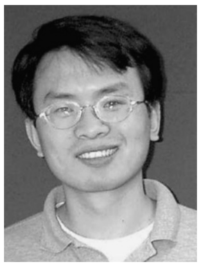


Kai Zhang received the Ph.D. degree in computer science from the Hong Kong University of Science and Technology in 2008. Then he joined the Life Sciences Division, Lawrence Berkeley National Laboratory in Berkeley, CA as a Postdoc Researcher. He is currently with NEC Laboratories America, Inc. in Princeton, NJ. His research interests include large scale machine learning, bionformatics and complex networks.



Zhengzhang Chen is currently a Researcher at NEC Laboratories America Inc., and an Adjunct Research Assistant Professor in the Department of Electrical Engineering and Computer Science at Northwestern University. He earned his Ph.D. in Computer Science from North Carolina State University in 2012, and his research interests include data mining, bioinformatics, graph algorithms, social computing, machine learning, and their applications.

Dequan Wang is currently an Undergraduate Student from the school of computer science, Fudan University. His research includes Computer Vision and Machine Learning.



Geoff Jiang received the B.S. and Ph.D. degrees in electrical and computer engineering from Beijing Institute of Technology, China, in 1993 and 1998, respectively. During 1998–2000, he was a Postdoctoral Fellow in computer engineering at Dartmouth College, New Hampshire. He is currently the Vice President of solution technology at NEC Laboratories America (NECLA) in Princeton, NJ. His current research focus is on distributed system, dependable and secure computing, system and information theory. He has published over 150 technical papers and also has over 70 patents granted or applied. He is an Associate Editor of the IEEE Security and Privacy magazine and has served in the program committees of many prestigious conferences.

Ivan Marsic received the Dipl.Ing. and M.S. degrees in computer engineering from the University of Zagreb, Croatia, and the Ph.D. degree in biomedical engineering from Rutgers University, New Brunswick, NJ, USA, in 1994. He is currently a Professor with the Department of Electrical and Computer Engineering, Rutgers University. His current research interests include sensor networks and machine learning for healthcare applications.