alignment on the training set will indicate a good alignment on the test set (Cristianini et al. 2002). On the other hand, $A_{K,y}$ is favorably associated with the generalization performance of a classifier (such as the Parzen window estimator) (Cristianini et al. 2002). Therefore, maximizing the alignment of the kernel with the ideal one provides a general and effective way for kernel design. Recently, a centralized alignment criterion was proposed in (Cortes, Mohri, and Rostamizadeh 2010), K and K',

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

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 (Cristianini et al. 2002). Sinha et al. presented a framework for computing sparse combination of the base kernels (Sinha and Belkin 2009). Cortes et al. show that the weighting in the maximal alignment kernel can be solved via quadratic programming (Cortes, Mohri, and Rostamizadeh 2010). In (Lanckriet et al. 2004), a semi-definite programming formulation was adopted to learn a kernel matrix $\tilde{K}$ that is maximally aligned with the ideal kernel. In (Zhu et al. 2004), an order constraint on the transformed eigenvalue is further considered. 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 (Zhu et al. 2004).

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. Therefore, these eigenvectors may not reveal useful structures for classification, and the base kernels they span can have low alignment with the target. 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 QCQP ($O(n^4)$) (Zhu et al. 2004) and SDP ($O(n^{4.5})$) (Lanckriet et al. 2004) is also quite demanding.

To solve these problems, we propose a new way for designing semi-supervised kernel. Besides using the eigenvectors from the original kernel matrix or graph Lapacian, 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. 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.

The rest of the paper is organized as follows. In section 2.1, we discuss the connection between kernel eigenvectors and class labels; Section 2.2 introduces the eigenfunction and its extrapolation via the Nystrom extension; Section 2.3 presents the complete algorithm; Section 2.4 analyzes the complexity of the proposed method. Section 3 discusses related methods. Section 4 compares the performance of our approach with a number of algorithms based on spectral transformations. The last section concludes the paper.

Proposed Method

Kernel target alignment is an important criterion widely used in semi-supervised kernel design (Cristianini et al. 2002). It states that higher alignment with the ideal kernel will indicate the existence of a good classifier with higher probabilities (Cristianini et al. 2002)(Cortes, Mohri, and Rostamizadeh 2010). It can be easily observed that the overall alignment of the mixed kernel depends directly on the individual alignment. For example, it has been shown that the optimized alignment between a mixture of kernel eigenvectors and the learning target is proportional to average of the individual alignment scores of these eigenvectors (Cristianini et al. 2002). In other words, base kernels with higher individual alignment to target will also lead to a higher overall alignment. 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 eigenstructures faithfully reflects 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.

Kernel Eigenvectors and Class Labels

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

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

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

Proposition 1 shows that non-zero eigenvectors of the ideal kernel correspond exactly to the classes labels (up to a scaling). For example, (Shi and Malik 2000) shows that the eigenvectors corresponding to the second smallest eigenvalue of the normalized graph Laplacian provides 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 its unsupervised nature; in comparison, class labels represent 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 higher target alignment. To achieve this goal, we need notions of eigenfunction and its extrapolation via the Nyström extension.

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 (Williams and Seeger 2000)

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

The standard numerical method to approximate the eigenfunctions and eigenvalues in (2) is to replace the integral with the empirical average (Fowlkes et al. 2004; Williams and Seeger 2000)

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

where $\mathbf{x}_i$, $_{i=1,2,\dots,q}$ is drawn from the distribution $f(\cdot)$. By choosing $\mathbf{z}$ also from $\mathbf{z} = \mathbf{x}_i$, $_{i=1,2,\dots,q}$, equation (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

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

This is known as the Nyström extension (Williams and Seeger 2001; Zhang, Tsang, and Kwok 2008; Zhang and Kwok 2010), 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, (4) is proportional to the projection of a test point computed in kernel PCA (Bengio et al. 2004). The approximation can be justified by examining the convergence of eigenvalues and eigenvectors as the number of examples increases (Shawe-Taylor and Williams 2003; Bengio et al. 2004).

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 $k = 1, 2, ..., c$, corresponding to the c classes, then we have

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

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 (see Appendix). 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.

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. However, the number of such eigenvectors is typically limited (bounded by the number of classes); on the other hand, in case the number of labeled sample is very limited, using the ideal kernel eigenvectors alone may not be sufficient. Therefore, it's safer to incorporate traditional kernel eigenvectors as well.

Suppose we have obtained a set of c extrapolated eigenvectors u_1,u_2,..,u_c, as well as a set of k eigenvectors v_1, v_2,

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

..., v_k from the empirical kernel matrix (or graph Lapacian). Then we want to learn the following kernel

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

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 literatures there has been various ways to compute the weights such as uniform weighting, independent alignment-based weighting, or the quadratic programming approach. We used the **alignf** procedure (Cortes, Mohri, and Rostamizadeh 2010) in our experiment. The optimal weighting scheme is not the focus of this paper, instead, we want to show that the quality of base kernels is the key to a successful alignment.

The learned kernel $\tilde{K}$ can be used directly in SVM (or SVR). 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 $Y = [\mathbf{y}_1, \mathbf{y}_2, ..., \mathbf{y}_c] \in \mathbf{R}^{l\times c}$.

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. Compute the extrapolated ideal kernel eigenvectors as $[u_1, u_2, ..., u_c] = D_n^{-1} K_{nl} Y$;
3. Compute the eigenvectors with dominant k eigenvalues of the kernel matrix or diminishing k eigenvalues of the (normalized) graph Laplacian, as $[v_1, v_2, ..., v_k]$;
4. Compute the weighting coefficients of the base eigenvectors $[u_1, u_2, ..., u_c, v_1, v_2, ..., v_k]$ using any of the existing weighting schemes;
5. Compute $\tilde{K} = \sum_{i=1}^{c} \alpha_i u_i u_i^\top + \sum_{j=1}^{k} \beta_j v_j v_j^\top$;
6. Apply $\tilde{K}$ for training and testing.

Complexity

Most steps in Algorithm 1 requires linear space and time w.r.t. c, k, and l, which equals number of classes, number of traditional kernel eigenvectors, and number of labeled samples, respectively. In step 3, a naive computation takes $O(n^2k)$ time, however, one can use the Nyström low-rank approximation technique to compute the dominant k eigenvectors of $n \times n$ kernel matrix with $O(nm^2)$ time, where $m \ll n$ is number of landmarks selected, see details in (Zhang, Tsang, and Kwok 2008). 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 and time needed is $O(nl)$. In step 6, the training takes empirically $O(l^{2.3})$ time using the libsvm package, and testing takes $O(pn + ln)$. In practice, we have $l, p \ll n$. Therefore, overall our algorithm has a linear time and space complexities.

Discussions

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 (Zhou et al. 2004) 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 estimated class label, S is the normalized kernel matrix $S = D^{-1/2} K D^{-1/2}$, and Y is the class label (unlabeled entries are filled with 0's). Zhou's 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 (Zhu, Ghahramani, and Lafferty 2003), the authors utilized the harmoic property

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

where f is the estimated label, K and D is the kernel matrix and degree matrix. 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). However, Zhu et al. use this property as a global constraint, and compute 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 Zhu's method in one task of wireless sensor localization.

Recently, a generalized Nyström method is proposed to learn a semi-supervised low-rank approximation of the kernel matrix (Zhang et al. 2008). There, the label information is utilized to learn a dictionary kernel defined only on a set of selected landmark points; in comparison, we directly extrapolate the class labels onto the whole data set via Nyström eigenfunction expansion.

Experiments

This section compares our method with a number of state-of-the-art algorithms for semi-supervised kernel design, for both classification and regression.

Classification

In this section, we compare the following methods for semi-supervised kernel design: (1) cluster kernel (Chapelle, Weston, and Scholkopf 2003), where $r(\cdot)$ is chosen as linear function $r(\lambda) = \lambda$; (2) diffusion kernel $r(\lambda) = \exp(-\lambda/\delta)$

Table 1: Classification Performance 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	**3.82±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	**6.64±1.27**	**6.56±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	**13.44±2.41**	18.51±4.66	19.08±2.05	19.32±1.89
BSI	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	**32.95±3.38**	**32.99±3.10**	42.02±2.89	33.58±2.83	34.85±2.75
COIL	0.01±0.001 (199.5)	0.11±0.005 (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	**9.14±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	**13.64±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	**23.42±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	**2.82±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	**1.98±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	**2.44±0.59**	3.89±1.57	3.85±0.97	5.79±1.06
usps27	0.58±0.004 (101.8)	0.91±0.006 (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	**1.21±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	**9.58±1.56**	14.59±1.49	14.08±2.04	15.64±2.91

(Kondor and Lafferty 2002); (3) maximal alignment kernel (Cristianini et al. 2002) using the top 0.1n eigenvectors from the kernel matrix; (4) our approach; (5) non-parametric graph kernel (Zhu et al. 2004) 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.

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 parameter 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. Method (1) and (5) use $10\%n$ diminishing eigenvectors from the normalized graph Lapacian; 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.

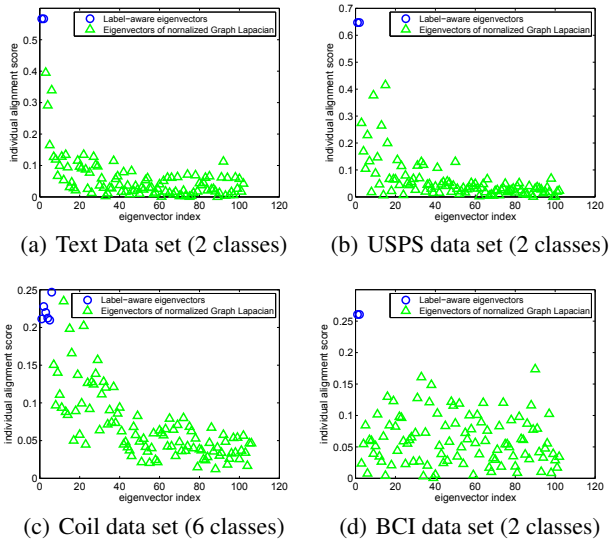
In Figure 1, we examine alignment score of the label-aware eigenvectors (blue circles) and those from the normalized Graph Lapacian[2]. Here the reported score is the average alignment between one eigenvector and all the c target variables. As can be seen, the label-aware eigenvectors almost always have higher or at least very similar alignment scores compared with the eigenvectors of the graph Lapacian.

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

Regression

In this section, we report empirical results of our algorithm in kernel based regression problems. The task is indoor location estimation using received signal strength(RSS) that a client device received from Wi-Fi access points (Yang, Pan, and Zheng 2000). We compare our result with Zhu's method (Zhu, Ghahramani, and Lafferty 2003). In particular, we have adopted the support vector regression (Smola and Scholkopf 2004) that works on the learned kernel in Algorithm 1. We have normalized the labels y_i's such that they scale in the range $[0, 1]$. We used 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$\{0.1, 1, 10, 100, 1000, 10000\}$.

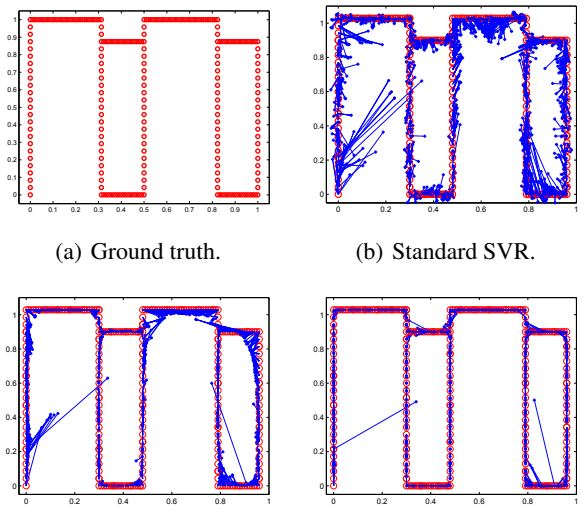
In Figure 2, we plot the regression results on the 2-D plane. Here red circles are the true coordinates, and blue dots are estimated ones. A line is connected between every pair of true and estimated points. As can be seen, our approach provides better localization results compared with Zhu's method. We have used the square root of the mean squared error to measure the regression quality The error of standard SVM is 2.5×10^{-3}; that of Zhu'e method is



(a) Text Data set (2 classes) (b) USPS data set (2 classes)

(c) Coil data set (6 classes) (d) BCI data set (2 classes)

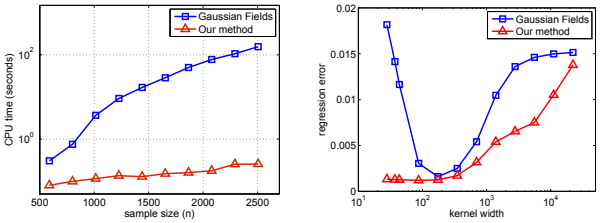
Figure 1: The individual target alignment score of label-aware base eigenvectors and the traditional kernel eigenvectors **on the unlabeled data**. For simplicity of visualization, here the reported score is the average alignment between one eigenvector and all the c target variables/classes.



(a) Ground truth. (b) Standard SVR.

(c) Gaussian Fields. (d) Our method.

Figure 2: Localization results by different methods. For each test point, a line is connected between the true and the estimated location/coordinate.



(a) Time versus sample size. (b) Error versus kernel width.

Figure 3: Gaussian Field method and our approach.

1.61×10^{-3}; while ours is around 1.19×10^{-3}. Our regression error is reduced by about 25% compared with Zhu's method, and more than 50% compared with the standard supervised SVR. In Figure 3(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 Zhu's method.In Figure 3(b), we plot the regression error of the two methods with regard to the Gaussian kernel width. As can be seen, our approach is more insensitive 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 stat-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 field method as an example for comparison because it has shown to provide stat-of-the-art results in this localization task (Yang, Pan, and Zheng 2000).

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, we pay more attention to the quality of the base kernel. In particular, we compute the label-aware eigenvectors via extending the ideal kernel eigenfunction. We show that our algorithm gives encouraging performance and scales linearly with sample size and dimension.

Appendix

Proposition 1 *Let the eigenvalue decomposition of* K^* *be* $K^*\boldsymbol{v}^* = \lambda^*\boldsymbol{v}^*$. *Since* K^* *only has* c *different rows (orthogonal to each other), it has rank* c *with* $n - c$ *zero eigenvalues. Note that the* i*th entry of* $\boldsymbol{v}^*$ *equals* $\frac{1}{\lambda^*}K^*(i,:)\boldsymbol{v}^*$, *and* K^* *has a block-wise constant structure. Therefore* $\boldsymbol{v}^*$ *is piecewise constant. Write* $\boldsymbol{v}^*$ *as* $[\underbrace{v_1,...,v_1}_{l_1}\underbrace{v_2,...,v_2}_{l_2},...,\underbrace{v_c,...,v_c}_{l_c}]'$. *Then the eigensystem becomes* $m_k v_k = \lambda^* v_k$ *for* $k = 1, 2, ..., c$. *Each equation leads to two conditions:* $\lambda^* = l_k$, *or* $v_k = 0$. *However, the former condition is infeasible for* $k = 1, 2, ..., C$, *since the size of different classes can be different. So one sets* λ^* *equal to one of the* ml_k*'s, i.e.,* $\lambda^* = l_{k_0}$, *and* $v_k = 0$ *for all* $k \neq k_0$. *There are* c *different ways to choose* k_0, *i.e.,* $k_0 = 1, 2, ..., c$. *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 normalize to 1).*

References

Belkin, M.; Niyogi, P.; and Sindhwani, V. 2006. Manifold regularization: A geometric framework for learning from labeled and unlabeled examples. *Journal of Machine Learning Research* 7.

Bengio, Y.; Delalleau, O.; Roux, N. L.; Paiement, J.-F.; Vincent, P.; and Ouimet, M. 2004. Learning eigenfunctions links spectral embedding and kernel pca. *Neural Computation* 16.

Chapelle, O., and Zien, A. 2005. Semi-supervised classification by low density separation. In *Proceedings of the 10th International Conference on Artificial Intelligence and Statistics*.

Chapelle, O.; Weston, J.; and Scholkopf, B. 2003. Cluster kernels for semi-supervised learning. In *Advances in Neural Information Processing Systems*.

Collobert, R.; Sinz, F.; Weston, J.; and Bottou, L. 2006. Large scale transductive svms. *Journal of Machine Learning Research* 7.

Cortes, C.; Mohri, M.; and Rostamizadeh, A. 2010. Two-stage learning kernel algorithms. In *Proceedings of the 27th Annual International Conference on Machine Learning*.

Cristianini, N.; Kandola, J.; Elissee, A.; and Shawe-Taylor, J. 2002. On kernel-target alignment. In *Advances in Neural Information Processing Systems*.

Fowlkes, C.; Belongie, S.; Chung, F.; and Malik, J. 2004. Spectral grouping using the nystrom method. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 26.

He, X.; Ji, M.; Zhang, C.; and Bao, H. 2011. A variance minimization criterion to feature selection using laplacian regularization. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 33.

Joachims, T. 1999. Transductive inference for text classification using support vector machines. In *Proceedings of the 16th Annual International Conference on Machine Learning*.

Kondor, R. I., and Lafferty, J. 2002. Diffusion kernels on graphs and other discrete input spaces. In *Proceedings of the 19th Annual International Conference on Machine Learning*.

Kulis, B.; Basu, S.; Dhillon, I.; and Mooney, R. 2005. Semi-supervised graph clustering: a kernel approach. In *Proceedings of the 22th Annual International Conference on Machine Learning*.

Kwok, J. T., and Tsang, I. W. 2003. Learning with idealized kernels. In *Proceedings of the 20th Annual International Conference on Machine Learning*.

Lanckriet, G.; Cristianini, N.; Bartlett, P.; Ghaoui, L. E.; and Jordan, M. 2004. Learning the kernel matrix with semidefinite programming. *Journal of Machine Learning Research* 5.

Li, Y., and Zhou, Z. 2011. Towards making unlabeled data never hurt. In *Proceedings of the 28th Annual International Conference on Machine Learning*.

Li, Z.; Liu, J.; and Tang, X. 2008. Pairwise constraint propagation by semidefinite programming for semi-supervised classification. In *Proceedings of the 25th Annual International Conference on Machine Learning*.

Melacci, S., and Belkin, M. 2011. Laplacian support vector machines trained in the primal. *Journal of Machine Learning Research* 12.

Shawe-Taylor, J., and Williams, C. 2003. The stability of kernel principal components analysis and its relation to the process eigenspectrum. In *Advances in Neural Information Processing Systems*.

Shi, J., and Malik, J. 2000. Normalized cuts and image segmentation. *IEEE Trasactions on Pattern Analysis and Machine Intelligence* 22.

Sinha, K., and Belkin, M. 2009. Semi-supervised learning using sparse eigenfunction bases. In *Advances in Neural Information Processing Systems*.

Smola, A., and Kondor, R. 2003. Kernels and regularization on graphs. In *In Conference on Learning Theory*.

Smola, A., and Scholkopf, B. 2004. A tutorial on support-vector regression. *Journal Statistics and Computing* 14.

Williams, C., and Seeger, M. 2000. The effect of the input density distribution on kernel-based classifiers. In *Proceedings of the 17rd Annual International Conference on Machine learning*.

Williams, C., and Seeger, M. 2001. Using the nystrom method to speed up kernel machines. In *Advances in Neural Information Processing Systems*.

Yang, Q.; Pan, S. J.; and Zheng, V. W. 2000. Estimating location using wi-fi. *IEEE Intelligent Systems* 23.

Zhang, K., and Kwok, J. 2010. Clustered Nyström method for large scale manifold learning and dimension reduction. *IEEE Trasactions on Neural Networks* 21.

Zhang, K.; Lan, L.; Liu, J.; Rauber, A.; and Moerchen, F. 2008. Inductive kernel low-rank decomposition with priors: A generalized nystrom method. In *Proceedings of the 29th International conference on machine learning*, 305–312.

Zhang, K.; Tsang, I.; and Kwok, J. T. 2008. Improved Nyström low-rank approximation and error analysis. In *International Conference on Machine Learning*, 1232–1239.

Zhou, X., and Belkin, M. 2011. Semi-supervised learning by higher order regularization. In *The 14th International Conference on Artificial Intelligence and Statistics*.

Zhou, D.; Bousquet, O.; Lal, T. N.; Weston, J.; and Scholkopt, B. 2004. Learning with local and global consistency. In *Advances in Neural Information Processing Systems*.

Zhu, X.; Kandola, J.; Ghahbramani, Z.; and Lafferty, J. 2004. Nonparametric transforms of graph kernels for semi-supervised learning. In *Advances in Neural Information Processing Systems*.

Zhu, X.; Ghahramani, Z.; and Lafferty, J. 2003. Semi-supervised learning using gaussian fields and harmonic functions. In *Proceedings of the 20th Annual International Conference on Machine Learning*.