Graph-based Semi-supervised Learning and Spectral Kernel Design

Rie Johnson, RJ Research Consulting, Tarrytown, NY 10591
Tong Zhang, Statistics Department, Rutgers University, Piscataway, NJ 08854

***Abstract*—We consider a framework for semi-supervised learning using spectral decomposition-based unsupervised kernel design. We relate this approach to previously proposed semi-supervised learning methods on graphs. We examine various theoretical properties of such methods. In particular, we present learning bounds and derive optimal kernel representation by minimizing the bound. Based on the theoretical analysis, we are able to demonstrate why spectral kernel design based methods can improve the predictive performance. Empirical examples are included to illustrate the main consequences of our analysis.**

***Index Terms*—Graph-based semi-supervised learning, Kernel design, Transductive learning**

I. Introduction

Spectral graph methods have been used both in clustering and in semi-supervised learning. This paper focuses on semi-supervised learning, where a classifier is constructed from both labeled and unlabeled training examples. Although previous studies showed that this class of methods work well for certain concrete problems (for example, see [2], [13], [17], [18]), there is no satisfactory theory demonstrating why, and under what circumstances, such methods should work. Moreover, from the previous studies, it is not clear what the exact relationship of graph-based semi-supervised learning is to the standard supervised kernel learning.

The purpose of this paper is to develop a more complete theoretical understanding for graph-based semi-supervised learning. In Theorem 3.1, we present a transductive formulation of kernel-learning on graphs which is equivalent to supervised kernel-learning. This kernel learning formulation includes some of the previously proposed graph semi-supervised learning methods as special cases. Consequently, we can view such graph-based semi-supervised learning methods as kernel design methods that utilize unlabeled data; the designed kernel is then used in the standard supervised learning setting. This insight allows us to prove useful results concerning the behavior of graph-based semi-supervised learning from a more general view of spectral kernel design. Similar spectral kernel design ideas also appeared in [5], [19]. However, they did not present a graph-based learning formulation (Theorem 3.1 in this paper); nor did they study the theoretical properties of such methods. We focus on two issues for graph kernel learning formulations based on Theorem 3.1. First, we establish the convergence of graph-based semi-supervised learning (when the number of unlabeled data points increases). Second, we obtain a learning bound, which can be used to compare the performance of different kernels. This analysis gives insight into what are good kernels, and why graph-based spectral kernel design is often helpful in various applications. Empirical examples are given to justify the theoretical analysis.

The paper is organized as follows. In Section II we briefly introduce supervised kernel learning formulations. In Section III, we establish a new equivalent formulation of supervised kernel learning on data graphs. This formulation forms the basis of our analysis presented in this paper. Based on this formulation, we show that one can formulate graph-based semi-supervised learning as kernel-design on graphs. The resulting algorithms will be investigated in the subsequent sections. Section IV develops the feature formulation of the graph-based semi-supervised learning algorithm, which we use to show that the algorithm converges when the number of graph nodes goes to infinity. Section V develops a transductive bound for graph-based learning, which we will use later to analyze the effect of spectral kernel design. Section VI presents a model for which spectral kernel design method introduced in the paper is effective. Finally, in Section VII, empirical results are used to validate various aspects of our theoretical results.

II. Supervised Kernel Methods

Consider the problem of predicting a real-valued output y based on its corresponding input vector x. In machine learning, our goal is to estimate a functional

Portions of this work were done when the authors were at IBM T.J. Watson Research Center and the second author was at Yahoo Inc.

Partially supported by NSF grant DMS-0706805

relationship $y \approx p(x)$ from a set of training examples. Usually the quality of a predictor $p(x)$ can be measured by a loss function $L(p(x), y)$.

In the standard machine learning formulation, we assume that the data (X, Y) are drawn from an unknown underlying distribution D. Our goal is to find $p(x)$ so that the expected true loss of p given below is as small as possible:

$$R(p(\cdot)) = E_{(X,Y)\sim D} L(p(X), Y),$$

where we use $E_{(X,Y)\sim D}$ to denote the expectation with respect to the true (but unknown) underlying distribution D. Typically, one needs to restrict the hypothesis function family size so that a stable estimate within the function family can be obtained from a finite number of samples. Let the training samples be $(X_1, Y_1), \ldots, (X_n, Y_n)$. We assume that the hypothesis function family that predicts y based on x can be specified with the following kernel method:

$$p(\alpha, x) = \sum_{i=1}^{n} \alpha_i k(X_i, x), \tag{1}$$

where $\alpha = [\alpha_i]_{i=1,\ldots,n}$ is a parameter vector that needs to be estimated from the data, and k is a symmetric positive kernel. That is, $k(a, b) = k(b, a)$, and the $n \times n$ Gram matrix $K = [k(X_i, X_j)]_{i,j=1,\ldots,n}$ is always positive semi-definite.

Definition 2.1: Let $\mathcal{H}_0 = \{\sum_{i=1}^{\ell} \alpha_i k(x_i, x) : \ell \in N, \alpha_i \in R\}$. $\mathcal{H}_0$ is an inner product space with norm defined as

$$\left\| \sum_i \alpha_i k(x_i, \cdot) \right\| = \left(\sum_{i,j} \alpha_i \alpha_j k(x_i, x_j) \right)^{1/2}.$$

Let $\mathcal{H}$ be the closure of $\mathcal{H}_0$ under the norm $\|\cdot\|$, which forms a Hilbert space, called the reproducing kernel Hilbert space (RKHS) of k. We denote the corresponding norm as $\|\cdot\|_{\mathcal{H}}$.

It is well-known and not difficult to check that the norm $\|\cdot\|_{\mathcal{H}}$ in Definition 2.1 is well-defined, and it defines an inner product. Further information on reproducing Hilbert spaces can be found in [14]. Definition 2.1 implies that $\forall p \in \mathcal{H}$:

$$p(x) = \langle p, k(x, \cdot) \rangle_{\mathcal{H}}, \tag{2}$$

where we use $\langle \cdot, \cdot \rangle_{\mathcal{H}}$ to denote the inner product in $\mathcal{H}$.

Given training data $\{(X_i, Y_i)\}_{i=1,\ldots,n}$, we can train a predictor $\hat{p} \in \mathcal{H}$ by minimizing the empirical loss. However, since the reproducing Kernel Hilbert space $\mathcal{H}$ can be large, it is often necessary to consider a regularized method in order to avoid overfitting:

$$\hat{p}(\cdot) = \arg \inf_{p \in \mathcal{H}} \left[\frac{1}{n} \sum_{i=1}^{n} L(p(X_i), Y_i) + \lambda \|p\|_{\mathcal{H}}^2 \right], \tag{3}$$

where $\lambda > 0$ is an appropriately chosen regularization parameter.

Although the minimization is in a Hilbert space $\mathcal{H}$, it is well-known that the solution of (3) can be represented as $\hat{p}(x) = \sum_{i=1}^{n} \hat{\alpha}_i k(X_i, x)$, and hence, (3) is equivalent to

$$\begin{aligned} \hat{p}(\cdot) &= \sum_{i=1}^{n} \hat{\alpha}_i k(X_i, \cdot), \\ [\hat{\alpha}_i] &= \arg \inf_{[\alpha_i] \in R^n} \left[\frac{1}{n} \sum_{i=1}^{n} + L \left(\sum_{j=1}^{n} \alpha_j k(X_i, X_j), Y_i \right) \right. \\ &\qquad \left. + \lambda \sum_{i,j=1}^{n} \alpha_i \alpha_j k(X_i, X_j) \right]. \end{aligned} \tag{4}$$

This method changes the possibly infinite dimensional problem (3) into an equivalent finite dimensional problem in terms of the n-dimensional vector α. Therefore we can solve the problem efficiently. In the literature, α is often solved through a different formulation (so-called dual formulation) which can be obtained using convex duality. The formulation is not important for our purpose since we only need to use (4) to define the estimator $\hat{\alpha}$.

III. Semi-supervised Learning on Graphs

In this section, we present an equivalence of supervised kernel learning to a specific semi-supervised formulation. Although this representation is implicit in the Gaussian process literature (for example, see [10]), to the best of the authors' knowledge, the general form of this representation is not well known in the machine learning community. In particular, this equivalence has not been explicitly stated in the previous studies on graph-based semi-supervised learning, although similar observations were made and discussed implicitly (see page 8 of [4], Section 2 of [20], and [1]). As we shall see later, the representation is critical for the purpose of obtaining a better understanding of graph-based semi-supervised learning. Therefore in spite of its simple proof and its close relationship to the more well-known representer theorem, we shall formally state it here as an independent theorem for future reference.

In our framework, the *data graph* consists of nodes that are the data points X_j. The edge connecting two nodes X_i and X_j is weighted by $k(X_i, X_j)$, where the kernel function $k(\cdot, \cdot)$ is defined over the graph nodes. The theorem essentially implies that a formulation of graph-based semi-supervised learning is equivalent to the supervised learning method which employs the same kernel. It establishes the transductive graph kernel learning formulation we will study in this paper. As a consequence, we can extend the graph (and the kernel matrix) by including extra points without changing the results on the existing points.

Theorem 3.1 (Graph Kernel Learning): Consider labeled data $\{(X_i, Y_i)\}_{i=1,\ldots,n}$ and unlabeled data X_j ($j = n+1, \ldots, m$). Consider real-valued vectors $f = [f_1, \ldots, f_m]^T \in R^m$, and the following semi-supervised learning method:

$$\hat{f} = \arg \inf_{f \in R^m} \left[\frac{1}{n} \sum_{i=1}^{n} L(f_i, Y_i) + \lambda f^T K^{-1} f \right], \quad (5)$$

where K is an $m \times m$ matrix with $K_{i,j} = k(X_i, X_j)$. Let $\mathcal{H}$ be the RKHS of k, and $\hat{p}$ be the solution of (3), then

$$\hat{f}_j = \hat{p}(X_j) \quad (j = 1, \ldots, m).$$

Proof: At the optimal solution of (5), we have the following first-order optimality equation (for example, see [12]):

$$\frac{1}{n} L_1'(\hat{f}_i, Y_i) + 2\lambda [K^{-1}\hat{f}]_i = 0 \quad (i = 1, \ldots, n),$$
$$2\lambda [K^{-1}\hat{f}]_j = 0 \quad (j = n+1, \ldots, m).$$

where $L_1'(a, b)$ denotes a sub-gradient of L with respect to the first component, and $[K^{-1}\hat{f}]_i$ denotes the i-th component of the m-dimensional vector $K^{-1}\hat{f}$. Now we can let $\hat{\alpha}_i = -L_1'(\hat{f}_i, Y_i)/(2\lambda n)$ for $i = 1, \ldots, n$, and $\hat{\alpha}_j = 0$ for $j = n+1, \ldots, m$. Substituting the representation into the first-order condition, we obtain a representation of $\hat{f}$ as:

$$\hat{f} = K\hat{\alpha}, \quad \text{s.t. } \hat{\alpha}_j = 0 \ (j = n+1, \ldots, m)$$

Using this representation in (5), we obtain the following equivalent optimization problem:

$$\hat{f} = K\hat{\alpha},$$
$$\hat{\alpha} = \arg \inf_{\alpha \in R^m} \left[\frac{1}{n} \sum_{i=1}^{n} L([K\alpha]_i, Y_i) + \lambda \alpha^T K \alpha \right]$$
$$\text{s.t. } \alpha_j = 0 \ (j = n+1, \ldots, m).$$

It is easy to check that this optimization problem is equivalent to (4) with $\hat{f}_j = [K\hat{\alpha}]_j = \hat{p}(X_j)$. ■

It is worth mentioning that in the theorem, while the kernel function $k(\cdot, \cdot)$ is typically defined everywhere on the input space, the Gram matrix K only needs to be defined on the graph. Note that K is always positive semi-definite. However, there are two scenarios that require special interpretation:

- K is not full rank (singular): the correct interpretation of $f^T K^{-1} f$ is $\lim_{\mu \to 0^+} f^T (K + \mu I_{m \times m})^{-1} f$, where $I_{m \times m}$ is the $m \times m$ identity matrix. If we start with a given kernel k and let $K = [k(X_i, X_j)]$, then a semi-supervised learning method of the form (5) is equivalent to the supervised method (3).
- $\mathcal{R} = K^{-1}$ is not full rank: in this case, we need an extension of (5), where we replace $L(f_i, Y_i)$ by $L(f_i + Bh_i, Y_i)$ with B the null-space of $\mathcal{R}$, and take K to be the pseudo-inverse of $\mathcal{R}$. In this case, the Bh component is unregularized. Similarly we should include an unregularized component in (3) and (4). For simplicity, we will not consider this case here.

Observe that if we start with a given kernel k and let $K = [k(X_i, X_j)]$, then the semi-supervised learning method in the form of (5) is equivalent to the supervised method (3). It follows that for a formulation like (5), one should utilize the unlabeled data to replace K by a kernel $\bar{K}$ in (5), or k by $\bar{k}$ in (4). There are various ways to achieve this. For example, one may use unlabeled data to derive better generalization performance bounds, which can then be used to select a kernel from a set of kernels to optimize the bound.

The effect of semi-supervised learning can also be achieved with the following two stage procedure:

- Use unlabeled data to design a kernel $\bar{k}(\cdot, \cdot)$.
- Replace $k(\cdot, \cdot)$ by $\bar{k}(\cdot, \cdot)$ in the supervised formula (4).

In other words, the benefit of unlabeled data in this setting is to construct a good kernel based on unlabeled data. At first, this seems strange as it is not clear why one has to design a kernel based on the unlabeled data. However, we will later show that under some assumptions, this is a sensible thing to do.

Some of the previous graph-based semi-supervised learning studies employ the same formulation (5) with K^{-1} replaced by the graph Laplacian operator $\mathcal{L}$. We only consider the normalized version here:

$$f^T \mathcal{L} f = \alpha f^T f + \frac{1}{2} \sum_{i,j=1}^{m} k(x_i, x_j) \left(\frac{f_i}{\sqrt{d_i}} - \frac{f_j}{\sqrt{d_j}} \right)^2$$
$$= f^T [(1+\alpha) I - D^{-1/2} K D^{-1/2}] f,$$

where $d_i = \sum_{j=1}^{m} k(x_i, x_j)$ ($i = 1, \ldots, m$), and $D = \operatorname{diag}(\{d_i\})$. The transform $D^{-1/2} K D^{-1/2}$ is to normalize the kernel K such that the eigenvalues are in $[0, 1]$. The parameter $\alpha > 0$ is used to ensure that the

Laplacian kernel $\bar{K} = ((1+\alpha)I - D^{-1/2}KD^{-1/2})^{-1}$ is positive definite (see [17]).

The equivalence of the graph Laplacian formulation and supervised kernel learning (with kernel matrix $\bar{K} = \mathcal{L}^{-1}$) was not explicitly discussed in the earlier studies. We will show that this equivalence is important for good theoretical understanding because it clarifies the role of unlabeled data and simplifies the analysis (as we will see later in the paper). Based on this formulation, our focus is to understand the behavior of different kernels, and why one kernel is preferred to another. Moreover, by treating graph-based supervised learning as unsupervised kernel-design (see Figure 1), we essentially consider a setting more general than graph Laplacian based methods.

Based on the above discussion, we shall focus on the following unsupervised kernel design-based semi-supervised learning. Although some of the previously proposed graph methods are variants of this algorithm, our analysis still provides useful insights.

In Figure 1, we consider a general formulation of semi-supervised learning on data graphs through spectral kernel design. As a special case, we can let $s_j = g(\mu_j)$ in Figure 1, where g is a rational function, and then $\bar{K} = g(K/m)K$. In this special case, we do not have to compute the eigen-decomposition of K. Therefore we obtain a simpler algorithm with the $(*)$ in Figure 1 replaced by

$$\bar{K} = g(K/m)K. \tag{6}$$

The algorithm is described in Figure 2. The function $g(\cdot)$ can be regarded as a filter function which modifies the spectral of the kernel.

As mentioned earlier, the idea of using spectral kernel design has appeared in [5] although they did not base their method on the graph formulation (5). In [19], the spectrum of a graph Laplacian was modified to maximize kernel alignment. Although the idea is related, the procedure proposed there is not directly comparable to ours. The semi-supervised learning methods described in Figure 1 or Figure 2 are effective only when $\hat{f}'$ is a better predictor than $\hat{f}$ in Theorem 3.1; in other words, when the new kernel $\bar{K}$ is better than K. In order to gain a good understanding of the behavior of the algorithms, we are interested in the case $m \to \infty$. Therefore in the next few sections, we investigate the following issues:

- The limiting behavior of $\hat{f}'$ in the algorithms as $m \to \infty$; that is, whether $\hat{f}'_j$ converges for each j.
- Generalization performance of (5).
- Optimal kernel design by minimizing generalization error, and its implications.
- Statistical models under which spectral kernel design-based semi-supervised learning is effective.

IV. Feature-space Formulations

We want to show that as $m \to \infty$, the semi-supervised algorithm in Figure 1 is well-behaved. That is, $\hat{f}'_j$ converges as $m \to \infty$. This is one of the most fundamental issues concerning the semi-supervised learning algorithm proposed in Section III. For example, related issues for the graph Laplacian have been investigated recently in [3], [6]–[8]. Their results depend on geometric properties of the graph Laplacian, and require the existence of a continuous Laplacian on a well-defined manifold, from which the data are drawn. Due to the more specific assumptions employed, their results cannot be applied to our problem. The analysis presented here relies only on some basic algebraic properties of graph learning. Therefore it applies to situations more general than the graph Laplacian studied earlier. However, our analysis cannot provide any geometric meaning for the limiting solution that the semi-supervised algorithm converges to.

Technically, the behavior at large m is easier to understand when we consider the feature space representation, which we shall introduce below. Note that this representation is given here only for the sake of analysis and not for computational purposes. It is well-known that kernel classifiers are equivalent to linear classifiers with data embedded into a high dimensional space $\mathcal{F}$ (possibly infinite dimensional), which we call feature space. There is a feature representation $\psi(x) \in \mathcal{F}$ associated with each data point x such that $k(x, x') = \psi(x)^T\psi(x')$, where we use the standard linear algebra notation on the feature space, and consider features in $\mathcal{F}$ to be column vectors. In this setting, each function $p(\cdot) \in \mathcal{H}$ can be regarded as a linear classifier on $\mathcal{F}$ with weight vector w, such that $p(x) = w^T\psi(x)$ for all x, and $\|p\|_{\mathcal{H}}^2 = w^Tw$.

It is not difficult to show that the condition $\|p\|_{\mathcal{H}}^2 = w^Tw$ implies other properties described above. Therefore for reference, we shall introduce the following definition of feature space.

Definition 4.1: A feature space $\mathcal{F}$ for $\mathcal{H}$ is a vector representation such that each function $p(\cdot) \in \mathcal{H}$ corresponds to a weight vector $w_p \in \mathcal{F}$ and $\|p\|_{\mathcal{H}}^2 = w_p^Tw_p$. For each data point x, we define its feature representation as $\psi(x) = w_{k(x,\cdot)}$.

It is not hard to see that such a feature space exists. Let B be a complete orthonormal basis for $\mathcal{H}$, then for each $p \in H$, its coordinate with respect to B forms a feature vector. One may also loosely regard $\mathcal{H}$ itself as the feature space. However, it is convenient to distinguish $\mathcal{F}$ from $\mathcal{H}$ so that we can work with standard linear algebra notation.

```
Input: labeled data $[(X_i, Y_i)]_{i=1,\ldots,n}$
        unlabeled data $X_j$ $(j = n+1, \ldots, m)$
        kernel function $k(\cdot, \cdot)$
        shrinkage factors $s_j \geq 0$ $(j = 1, \ldots, m)$
Output: predictive values $\hat{f}'_j$ on $X_j$ $(j = 1, \ldots, m)$

Form the kernel matrix $K = [k(X_i, X_j)]$ $(i, j = 1, \ldots, m)$
Compute the kernel eigen-decomposition:
    $K = m \sum_{j=1}^{m} \mu_j v_j v_j^T$, where $(\mu_j, v_j)$ are eigenpairs of $K$ $(v_j^T v_j = 1)$
Modify the kernel matrix as:
    $\bar{K} = m \sum_{j=1}^{m} s_j \mu_j v_j v_j^T$        $(*)$
Compute
    $\hat{f}' = \arg\inf_{f \in R^m} \left[\frac{1}{n} \sum_{i=1}^{n} L(f_i, Y_i) + \lambda f^T \bar{K}^{-1} f\right]$.
```

Fig. 1. Spectral kernel design based semi-supervised learning on graph

```
Input: labeled data $[(X_i, Y_i)]_{i=1,\ldots,n}$
        unlabeled data $X_j$ $(j = n+1, \ldots, m)$
        kernel function $k(\cdot, \cdot)$
        shrinkage function $g(\cdot)$
Output: predictive values $\hat{f}'_j$ on $X_j$ $(j = 1, \ldots, m)$

Form the kernel matrix $K = [k(X_i, X_j)]$ $(i, j = 1, \ldots, m)$
Compute
    $\hat{f}' = \arg\inf_{f \in R^m} \left[\frac{1}{n} \sum_{i=1}^{n} L(f_i, Y_i) + \lambda f^T K^{-1} g(K/m)^{-1} f\right]$.
```

Fig. 2. Spectral filter design based semi-supervised learning on graph

Using the feature space representation, equation (3) is equivalent to

$$\begin{aligned} \hat{p}(x) =& \hat{w}^T \psi(x) \\ \hat{w} =& \arg\inf_{w \in \mathcal{F}} \left[\frac{1}{n} \sum_{i=1}^{n} L(w^T \psi(X_i), Y_i) + \lambda w^T w\right]. \end{aligned} \tag{7}$$

In this setting, a change of kernel can be regarded as a change of feature space. In particular, let S be a positive semi-definite matrix on $\mathcal{F}$. We may consider the following estimation method:

$$\begin{aligned} \hat{p}(x) =& \hat{w}^T S^{1/2} \psi(x) \\ \hat{w} =& \arg\inf_{w \in \mathcal{F}} \left[\frac{1}{n} \sum_{i=1}^{n} L(w^T S^{1/2} \psi(X_i), Y_i) + \lambda w^T w\right], \end{aligned} \tag{8}$$

which changes the feature vectors from $\psi(x)$ to $\psi'(x) = S^{1/2}\psi(x)$. It is easy to see that this method is equivalent to a change of kernel from $k(x, x') = \psi(x)^T \psi(x')$ to $\psi'(x)^T \psi'(x') = \psi(x)^T S \psi(x')$.

Our goal is to construct S so that (8) is equivalent to the graph-based semi-supervised learning methods in Section III. The re-formulation is useful since it is more convenient to study the asymptotic behavior of the S operator when $m \to \infty$ under the feature space representation (instead of the graph representation). This is because the graphs in Section III grow when m increases, while the feature space $\mathcal{F}$ remains the same.

The analysis given in this section is closely related to kernel principal component analysis. We still consider data X_i for $i = 1, \ldots, m$. let $\Psi = [\psi(X_1), \ldots, \psi(X_m)]$. Using this notation, the kernel matrix on X_i $(i = 1, \ldots, m)$ can be represented as $K = \Psi^T \Psi$. We shall design the matrix S in (8) based on the spectral decomposition of K. It can be seen that the spectral decomposition of $\Psi\Psi^T$ is closely related to that of K: if $\mu > 0$, then (μ, v) is an eigen-pair of $\Psi^T\Psi$ implies that $(\mu, \Psi v/\sqrt{\mu})$ is an eigen-pair of $\Psi\Psi^T$.

Lemma 4.1: Consider $K = \Psi^T\Psi$, where $\Psi = [\psi(X_1),\ldots,\psi(X_m)] \in \mathcal{F}^m$. Let $K = m\sum_j \mu_j v_j v_j^T$ be its spectral-decomposition with eigenvalues $m\mu_j > 0$ ($v_j^T v_j = 1$). Let $u_j = \Psi v_j/\sqrt{m\mu_j}$, then we have spectral decomposition $\Psi\Psi^T = m\sum_j \mu_j u_j u_j^T$. Let $S = \sum_{j=1}^m s_j u_j u_j^T$ and $\bar{K} = m\sum_{j=1}^m s_j \mu_j v_j v_j^T$, then $\bar{K}_{i,j} = \psi(X_i)^T S \psi(X_j)$.

Proof: Observe that $Kv_j = m\mu_j v_j$ implies that $(\Psi\Psi^T)\Psi v_j = \Psi K v_j = m\mu_j \Psi v_j$, which implies that u_j is an eigenvector of $\Psi\Psi^T$ with eigenvalue $m\mu_j$. We also have the following decomposition:

$$\begin{aligned}(\Psi\Psi^T)^2 =& \Psi K \Psi^T = m\sum_j \mu_j(\Psi v_j)(v_j^T\Psi^T)\\ =& m^2\sum_j \mu_j^2 u_j u_j^T.\end{aligned}$$

This implies that we have the spectral decomposition $\Psi\Psi^T = m\sum_j \mu_j u_j u_j^T$. Now we use $Kv_j = m\mu_j v_j$ to obtain:

$$\begin{aligned}\bar{K} =& m\sum_{j=1}^m s_j\mu_j v_j v_j^T = \sum_{j=1}^m s_j(Kv_j)(Kv_j)^T/(m\mu_j)\\ =& \sum_{j=1}^m s_j(\Psi^T\Psi v_j)(\Psi^T\Psi v_j)^T/(m\mu_j)\\ =& \sum_{j=1}^m s_j(\Psi^T u_j)(\Psi^T u_j)^T = \Psi^T S\Psi.\end{aligned}$$

This proves the second part of the lemma. ■

Using this lemma, we can obtain the feature space equivalence of Figure 1 and Figure 2. They are given in Figure 3 and Figure 4 respectively.

Theorem 4.1: Let $k(x,x') = \psi(x)^T\psi(x')$. Then Figure 2 and Figure 4 are equivalent: $\hat{f}'_j = \hat{p}'(X_j)$ ($j = 1,\ldots,m$). Moreover, if in Figure 1 and Figure 3, we use eigenvectors $u_j = \Psi v_j/\sqrt{m\mu_j}$, then the two algorithms are equivalent: $\hat{f}'_j = \hat{p}'(X_j)$ ($j = 1,\ldots,m$).

Proof: The first half of the theorem follows from the second half. To see this, we may let $s_j = g(\mu_j)$ and $u_j = \Psi v_j/\sqrt{m\mu_j}$. Hence by Lemma 4.1, we have $S = g(\Psi\Psi^T/m) = \sum_j g(\mu_j)u_j u_j^T = \sum_j s_j u_j u_j^T$ in Figure 4. This is consistent with the definition of s_j in Figure 2.

Therefore we only need to prove the second part. Based on the discussion after (8), we know that the solution of Figure 3 is equivalent to the solution of (4) with the kernel $k(x,x')$ replaced by $\bar{k}(x,x') = \psi(x)^T S\psi(x')$. By Theorem 3.1, on the data-graph, the solution is given by (5) with the kernel Gram matrix K replaced by

$$\begin{aligned}\bar{K} =& \Psi^T S\Psi = \sum_j s_j \Psi^T u_j u_j^T \Psi\\ =& \sum_j s_j(Kv_j/\sqrt{m\mu_j})(Kv_j/\sqrt{m\mu_j})^T\\ =& m\sum_j s_j\mu_j v_j v_j^T,\end{aligned}$$

which is consistent with the definition in Figure 1. This proves the second part of the theorem. ■

From the reformulation of Figure 2 as Figure 4 (and Figure 1 as Figure 3), we can easily understand the asymptotic behavior of these algorithms when $m \to \infty$. In this case, we just replace $\Psi\Psi^T/m = \frac{1}{m}\sum_{j=1}^m \psi(X_j)\psi(X_j)^T$ by $\mathbf{E}_X\psi(X)\psi(X)^T$. The spectral decomposition of $\mathbf{E}_X\psi(X)\psi(X)^T$ corresponds to the feature space PCA. It is clear that if S converges, then the feature space algorithm Figure 4 also converges, which immediately implies the convergence of the solution of Figure 2. We state the following result, which is self-evident. It shows that under mild conditions, the semi-supervised learning methods in Figure 4 and in Figure 2 are well behaved when $m \to \infty$.

Theorem 4.2: Consider a sequence of data $X_1, X_2, \ldots$ drawn from a distribution, with only the first n points labeled. Assume when $m \to \infty$, $\sum_{j=1}^m \psi(X_j)\psi(X_j)^T/m$ converges to $\mathbf{E}_X\psi(X)\psi(X)^T$ almost surely, and $g(\cdot)$ is a continuous function in the spectral range of $\mathbf{E}_X\psi(X)\psi(X)^T$. Then the following claims hold for Figure 4:

- S converges almost surely to $g(\mathbf{E}_X\psi(X)\psi(X)^T)$.
- $\bar{\psi}(x)$ converges almost surely to $g(\mathbf{E}_X\psi(X)\psi(X)^T)^{1/2}\psi(x)$.
- $\hat{p}'$ converges almost surely.
- With kernel $k(x,x') = \psi(x)^T\psi(x')$ in Figure 2, $\hat{f}_j = \hat{p}'(X_j)$ converges almost surely.

The theorem says that in the limit, the semi-supervised learning procedure is well-behaved in that the solution converges to a well-defined solution in (8) with a well-defined S. However, the geometric meaning of the operator S is not examined here. The assumption that $\sum_{j=1}^m \psi(X_j)\psi(X_j)^T/m$ converges to $\mathbf{E}_X\psi(X)\psi(X)^T$ almost surely is rather mild. It is essentially a consequence of the strong law of large numbers on vector spaces. Note also that statements similar to those of Theorem 4.2 hold for Figure 3 and Figure 1 as well. In this case, S converges if the eigenvectors u_j converge and the shrinkage factors s_j are bounded.

```
Input: labeled data $[(X_i, Y_i)]_{i=1,\ldots,n}$
       unlabeled data $X_j$ $(j = n+1, \ldots, m)$
       feature map $\psi(x) \in \mathcal{F}$
       shrinkage factors $s_j \geq 0$ $(j = 1, \ldots, m)$
Output: predictive function $\hat{p}'(x)$

Let $\Psi = [\psi(X_1), \ldots, \psi(X_m)]$
Compute the feature-space eigen-decomposition:
  $\Psi\Psi^T/m = \sum_{j=1}^m \mu_j u_j u_j^T$, where $(\mu_j, u_j)$ are eigenpairs of $\Psi\Psi^T$ $(u_j^T u_j = 1)$
Define
  $S = \sum_{j=1}^m s_j u_j u_j^T$, and $\bar{\psi}(x) = S^{1/2}\psi(x)$
Solve
  $\hat{w}' = \arg\inf_{w \in \mathcal{F}} \left[\frac{1}{n}\sum_{i=1}^n L(w^T\bar{\psi}(X_i), Y_i) + \lambda w^T w\right]$
Let $\hat{p}'(x) = \hat{w}'^T\bar{\psi}(x)$
```

Fig. 3. Spectral kernel design based semi-supervised learning on feature space

```
Input: labeled data $[(X_i, Y_i)]_{i=1,\ldots,n}$
       unlabeled data $X_j$ $(j = n+1, \ldots, m)$
       feature map $\psi(x) \in \mathcal{F}$
       shrinkage factors $s_j \geq 0$ $(j = 1, \ldots, m)$
Output: predictive function $\hat{p}'(x)$

Let $\Psi = [\psi(X_1), \ldots, \psi(X_m)]$
Define
  $S = g(\Psi\Psi^T/m)$ , and $\bar{\psi}(x) = S^{1/2}\psi(x)$
Solve
  $\hat{w}' = \arg\inf_{w \in \mathcal{F}} \left[\frac{1}{n}\sum_{i=1}^n L(w^T\bar{\psi}(X_i), Y_i) + \lambda w^T w\right]$
Let $\hat{p}'(x) = \hat{w}'^T\bar{\psi}(x)$
```

Fig. 4. Spectral filter design based semi-supervised learning on feature space

V. Generalization Analysis on Graphs

In this section, we study the generalization behavior of the graph-based semi-supervised learning algorithm (5), and use it to compare different kernels. We will then use this bound to justify the kernel design method given in Section III.

To measure the sample complexity, we consider m points (X_j, Y_j) for $i = 1, \ldots, m$. We randomly pick n distinct integers $i_1, \ldots, i_n$ from $\{1, \ldots, m\}$ uniformly (sample without replacement), and regard it as the n labeled training data. We obtain predictive values $\hat{f}_j$ on the graph using the semi-supervised learning method (5) with the labeled data, and test it on the remaining data (that are not included in the training data). We are interested in the average predictive performance over all random draws.

For clarity, we use a simple complexity analysis and compare expected generalization behavior. We focus on revealing the right quantities that determine the learning complexity, instead of trying to obtain the tightest and most general bound. The proof is given in the Appendix.

Theorem 5.1: Consider (X_j, Y_j) for $i = 1, \ldots, m$. Assume that we randomly pick n distinct integers $i_1, \ldots, i_n$ from $\{1, \ldots, m\}$ uniformly (sample without replacement), and denote it by Z_n. Let $\hat{f}(Z_n)$ be the semi-supervised learning method (5) using training data in Z_n:

$$\hat{f}(Z_n) = \arg\inf_{f \in R^m} \left[\frac{1}{n}\sum_{i \in Z_n} L(f_i, Y_i) + \lambda f^T K^{-1} f\right].$$

If $|L_1'(p, y)| = |\frac{\partial}{\partial p} L(p, y)| \leq \gamma$, and $L(p, y)$ is convex with respect to p, then we have

$$\mathbf{E}_{Z_n} \frac{1}{m-n} \sum_{j \notin Z_n} L(\hat{f}_j(Z_n), Y_j)$$
$$\leq \inf_{f \in R^m} \left[\frac{1}{m} \sum_{j=1}^m L(f_j, Y_j) + \lambda f^T K^{-1} f + \frac{\gamma^2 \mathrm{tr}(K)}{2\lambda nm} \right].$$

The complexity term in Theorem 5.1 relies on the quantity $\mathrm{tr}(K/\lambda n)$, which has appeared previously in the literature on related problems (for example, [9]). We should note that the bound in Theorem 5.1 is not necessarily optimal. For example, for the least squares method or logistic regression, a more refined quantity $\mathrm{tr}((K + \lambda n I)^{-1} K)$ was suggested in [16]. It is shown that for many problems, the refined quantity leads to the optimal rate of convergence. However, the bound in Theorem 5.1 is simpler and easier to work with. We are mainly interested in whether we can use it to explain the effectiveness of kernel design methods, and what are the implications of the bound. As we shall show in Section VII, the bound leads to observable consequences in practical applications. Therefore the analysis given here is adequate for our purpose (although refinements are possible).

The bound in Theorem 5.1 depends on the regularization parameter λ in addition to the kernel K. In order to compare different kernels, we shall compare bounds with the optimal λ for each K. That is, in addition to minimizing f, we also minimize over λ on the right hand of the bound. Note that in practice, it is often possible to find a near-optimal λ through cross validation when the amount of training data is not too small. This implies that assuming the optimal λ in the bound is reasonable for many practical problems. With optimal λ, we obtain:

Corollary 5.1: Under the conditions of Theorem 5.1, assume that we are given the optimal λ. Then:

$$\mathbf{E}_{Z_n} \frac{1}{m-n} \sum_{j \notin Z_n} L(\hat{f}_j(Z_n), Y_j)$$
$$\leq \inf_{f \in R^m} \left[\frac{1}{m} \sum_{j=1}^m L(f_j, Y_j) + \frac{\gamma}{\sqrt{2n}} \sqrt{R(f,K)} \right],$$

where

$$R(f,K) = \mathrm{tr}(K/m)\, f^T K^{-1} f$$

is the complexity of f with respect to kernel K.

If we define $\bar{K}$ as in Figure 1, then the complexity of a function f with respect to $\bar{K}$ is given by

$$R(f,\bar{K}) = \left(\sum_{j=1}^m s_j \mu_j \right) \left(\sum_{j=1}^m \alpha_j^2 / (s_j \mu_j) \right).$$

If we believe that a good approximate target function f can be expressed as $f = \sum_j \alpha_j v_j$ with $|\alpha_j| \leq \beta_j$ for some known β_j, then based on this belief, the optimal choice (according to the bound) of the shrinkage factor becomes $s_j = \beta_j / \mu_j$. That is, we use a kernel

$$\bar{K} = \sum_j \beta_j v_j v_j^T,$$

where v_j are normalized eigenvectors of K. In this case, we have $R(f,\bar{K}) \leq (\sum_j \beta_j)^2$. The eigenvalues of the optimal kernel is thus independent of K, but depends only on the spectral coefficient's range β_j of the approximate target function.

Since there is no reason to believe that the eigenvalues μ_j of the original kernel K are proportional to the target spectral coefficient range, if we have some guess of the spectral coefficients of the target, then one may use this knowledge to obtain a better kernel. This justifies why spectral kernel design-based algorithms can be potentially helpful (when we have some information on the target spectral coefficients). In practice, it is usually difficult to have a precise guess of β_j. However, for many applications, we observe in practice that the eigenvalues of kernel K decays slower than that of the target spectral coefficients. In this case, our analysis implies that we should use an alternative kernel with faster eigenvalue decay: for example, using K^2 instead of K. This has a dimension reduction effect. That is, we effectively project the data into the principal components of data. The intuition why this helps is also quite clear: if the dimension of the target function is small (spectral coefficient decays fast), then we should project data to those dimensions by reducing the remaining noisy dimensions (corresponding to fast kernel eigenvalue decay).

In the next section, we use a statistical model to illustrate why in practical problems, the spectral coefficients of the target function often decay faster than the eigenvalues of a natural kernel K. In essence, this is due to the fact that the input vector is often corrupted with small amounts of noise. Such noise does not significantly affect the target spectral coefficients, but will flattens the eigenvalues of K.

VI. Spectral analysis

We provide a justification of why spectral coefficients of the target function often decay faster than the eigenvalues of a natural kernel K. In essence, this is due to the fact that an input vector X is often corrupted with noise. Together with results in the previous section, we know that in order to achieve optimal performance, we may use a kernel with faster eigenvalue decay.

In this section we consider the feature space representation. We show that when the input feature is corrupted with small amounts of random noise, then the eigenvalues of K will become flatter, while the spectral coefficients of a reasonable target function will be less affected.

A. Input noise model

We consider a statistical model using the feature space notation in Section IV. For simplicity, we assume that $\psi(x) = x$. The model we consider here assumes that noise is added to the feature vector x. It is worth mentioning that although this model is justifiable for the linear kernel $k(x, x') = x^T x$, it may not be the most appropriate model for nonlinear kernels. However, we believe the analysis still provides useful insights for such kernels, although a more complete analysis requires further investigation.

We consider a two-class classification problem in R^∞ (with the standard 2-norm inner-product), where the label $Y = \pm 1$. We first start with a noise free model, where the data can be partitioned into p clusters. Each cluster ℓ is composed of a single center point $\bar{x}_\ell$ (having zero variance) with label $\bar{y}_\ell = \pm 1$. In this model, assume that the centers are well separated so that there is a weight vector w_* such that $w_*^T w_* < \infty$ and $w_*^T \bar{x}_\ell = \bar{y}_\ell$. Without loss of generality, we may assume that $\bar{x}_\ell$ and w_* belong to a p-dimensional subspace V_p. Let $V_p^\perp$ be its orthogonal complement.

Assume now that the observed input data are corrupted with noise. We first generate a center index ℓ, and then noise δ (which may depend on ℓ). The observed input data is the corrupted data $X = \bar{x}_\ell + \delta$, and the observed output is $Y = w_*^T \bar{x}_\ell$. In this model, let $\ell(X_i)$ be the center corresponding to X_i, the observation can be decomposed as: $X_i = \bar{x}_{\ell(X_i)} + \delta(X_i)$, and $Y_i = w_*^T \bar{x}_{\ell(X_i)}$. Given noise δ, we decompose it as $\delta = \delta_1 + \delta_2$ where δ_1 is the orthogonal projection of δ in V_p, and δ_2 is the orthogonal projection of δ in $V_p^\perp$. We assume that δ_1 is a small noise component; the component δ_2 can be large but has small variance in every direction.

B. Analysis

Under the model in Section VI-A, we have the following result.

Theorem 6.1: Consider the data generation model in Section VI-A, with observation $X = \bar{x}_\ell + \delta$ and $Y = w_*^T \bar{x}_\ell$. Assume that δ is conditionally zero-mean given ℓ: $\mathbf{E}_{\delta|\ell}\delta = 0$. Let $\mathbf{E}XX^T = \sum_j \mu_j u_j u_j^T$ be the spectral decomposition with decreasing eigenvalues μ_j ($u_j^T u_j = 1$). Then the following claims are valid:

- Let $\sigma_1^2 \geq \sigma_2^2 \geq \cdots$ be the eigenvalues of the noise covariance matrix $\mathbf{E}\delta_2\delta_2^T$, then $\mu_j \geq \sigma_j^2$.
- If $\|\delta_1\|_2 \leq b/\|w_*\|_2$, then $|w_*^T X_i - Y_i| \leq b$.
- If $w_*^T(\mathbf{E}\,\bar{x}_\ell\bar{x}_\ell^T)^{-t}w_* < \infty$ for some $t \geq 0$, then $\sum_{j\geq 1}(w_*^T u_j)^2\mu_j^{-t} \leq w_*^T(\mathbf{E}\,\bar{x}_\ell\bar{x}_\ell^T)^{-t}w_*$.

Proof: The conditional zero-mean of noise implies that

$$\mathbf{E}XX^T = \mathbf{E}\bar{x}_\ell\bar{x}_\ell^T + \mathbf{E}\delta\delta^T.$$

Given an arbitrary j dimensional subspace V, by the minimax principle for intermediate eigenvalues, we have the following inequality: $\mu_j \geq \inf_{v\in V} v^T\mathbf{E}XX^T v/v^T v$. Now we can take V to be the j-dimensional subspace spanned by the largest j eigenvectors of $\mathbf{E}\delta_2\delta_2^T$. Since $\delta_2 \in V_p^\perp$, it is clear that $V \in V_p^\perp$; therefore $\forall v \in V$, $v^T\delta = v^T\delta_2$ and $v^T\bar{x}_\ell = 0$. We thus have $\forall v \in V$:

$$v^T(\mathbf{E}XX^T)v = \mathbf{E}(v^T\delta_2)^2 \geq v^T v\sigma_j^2.$$

This implies the first claim: $\mu_j \geq \sigma_j^2$.

We also have: $|w_*^T X_i - Y_i| = |w_*^T\bar{x}_{\ell(X_i)} - Y_i + w_*^T\delta(X_i)| = |w_*^T\delta_1(X_i)| \leq \|w_*\|_2\|\delta_1(X_i)\|_2 \leq b$. This proves the second claim.

Using the following equations

$$\begin{aligned}\sum_{j\geq 1}(w_*^T u_j)^2/\mu_j^t =& w_*^T(\mathbf{E}XX^T)^{-t}w_* \\ =& w_*^T(\mathbf{E}\bar{x}_\ell\bar{x}_\ell^T + \mathbf{E}\delta\delta^T)^{-t}w_* \\ \leq& w_*^T(\mathbf{E}\bar{x}_\ell\bar{x}_\ell^T)^{-t}w_*,\end{aligned}$$

we obtain the third claim. ■

In order to relate the above theorem to kernel design methods on graphs, we may use the following simple result.

Proposition 6.1: Consider m points $X_1, \ldots, X_m$. Let $\Psi = [X_1, \ldots, X_m]$, $K = \Psi^T\Psi$, and let its spectral decomposition be $K = m\sum_j \mu_j v_j v_j^T$. Let $f_i = w_*^T X_i$, and let $f = \sum_j \alpha_j v_j$, then $\alpha_j = \sqrt{m\mu_j}w_*^T u_j$, where $u_j = \Psi v_j/\sqrt{m\mu_j}$.

Proof: We have $\alpha_j = f^T v_j = w_*^T\Psi v_j = \sqrt{m\mu_j}w_*^T u_j$. ■

This proposition implies that if we assume that asymptotically $\frac{1}{m}\sum_{i=1}^m X_i X_i^T \to \mathbf{E}XX^T$, then we have the following consequences from Theorem 6.1:

- $f_i = w_*^T X_i$ is a good approximate target when b is small. In particular, if $b < 1$, then this function always gives the correct class label.

- The spectral coefficients α_j of f decays as
$$\frac{1}{m}\sum_{j=1}^{m} \alpha_j^2/\mu_j^{1+t} \leq w_*^T(\mathbf{E}\bar{x}_\ell \bar{x}_\ell^T)^{-t} w_*.$$
That is, on average, target coefficients α_j decay at least as fast as the $\frac{1+t}{2}$-th power of the eigenvalues μ_j of K/m, if $w_*^T(\mathbf{E}\bar{x}_\ell \bar{x}_\ell^T)^{-t} w_*$ is bounded. Therefore α_j decays faster than μ_j (on average) if $t > 1$.
- The eigenvalue μ_j can decay slowly when the noise spectral decays slowly: $\mu_j \geq \sigma_j^2$. This slow decay is caused by noise in the feature vectors.

In summary, our analysis implies that if the clean data is well behaved, in the sense that we can find a target weight vector w_* such that $w_*^T(\mathbf{E}\bar{x}_\ell \bar{x}_\ell^T)^{-t} w_*$ is bounded for some $t > 1$, then when the observed data are corrupted with noise, we can find a good approximate target function f with spectral coefficients decaying faster than the eigenvalues of the kernel matrix.

C. *Spectral kernel design*

We showed that if the input data is corrupted with noise, then the spectral coefficient of the target function is likely to decay faster than that of the original kernel. It is thus helpful to use a kernel with faster spectral decay. For example, instead of using K, we may use K^2. However, it is may not be easy to estimate the exact decay rate of the target spectral coefficients. In practice, one may use cross validation to optimize the kernel. Another approach is to optimize a learning bound (e.g. Theorem 5.1), which may lead to semi-definite programming formulations [9].

A kernel with fast spectral decay projects the data into the most prominent principal components. Therefore in this paper, we are interested in designing kernels which can achieve a dimension reduction effect. Although one may use direct eigenvalue computation, an alternative is to use a filter function $g(K/m)K$, as in Figure 2. For example, we may consider normalized kernel such that $K/m = \sum_j \mu_j u_j u_j^T$ where $0 \leq u_j \leq 1$. A standard normalization method is to use $D^{-1/2}KD^{-1/2}$, where D is the diagonal matrix with each entry corresponding to the row sums of K.

It follows that $g(K/m)K = m\sum_j g(\mu_j)\mu_j u_j u_j^T$. We are interested in a function g such that $g(\mu)\mu \approx 1$ when $\mu \in [\alpha, 1]$ for some α, and $g(\mu)\mu \approx 0$ when $\mu < \alpha$ (where α is close to 1). One such function is to let $g(\mu)\mu = (1-\alpha)/(1-\alpha\mu)$. This is the function used in various graph Laplacian formulations with normalized Gaussian kernel as the initial kernel K. For example, see [17]. The function $g(\mu)$ is plotted against $1-\mu$ in Figure 5 (with $\alpha = 0.9, 0.99, 0.999$). Our analysis suggests that it is the dimension reduction effect of this function that is important, rather than the connection to the graph Laplacian. As we shall see in the empirical examples, other kernels such as K^2, which achieve similar dimension reduction effect (but have nothing to do with the graph Laplacian), also improve performance.

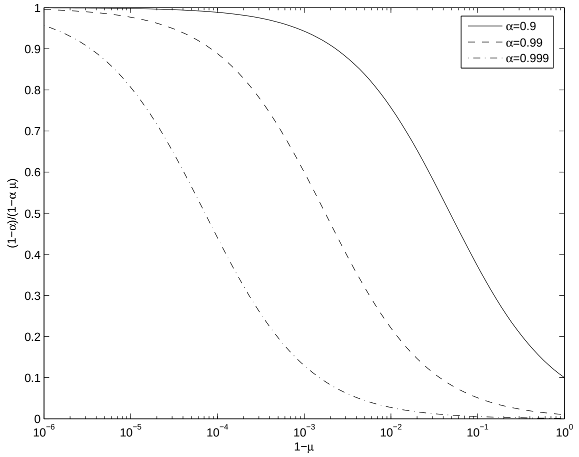

Fig. 5. $(1-\alpha)/(1-\alpha\mu)$ versus $1-\mu$.

VII. Experiments

This section uses empirical examples to demonstrate some consequences of our theoretical analysis. We shall use the MNIST data set and the 20newsgroup data set, both of which are commonly used by graph-based semi-supervised learning researchers.

A. *Data*

The MNIST data set[1] consists of hand-written digit image data (representing 10 classes, from digit “0” to “9”). The 20newsgroup data set consists of documents from 20 newsgroups (representing 20 classes ranging over a variety of topics – computer hardware, sports, and so on). In pre-processing, we removed the header lines (subjects, newsgroup names, senders, and so forth) and common stop words. We use the standard TFIDF term weighting to represent data points. On both data sets, we randomly draw $m = 2000$ samples, and then regard $n = 100$ of them as labeled data, and the remaining $m - n = 1900$ as unlabeled test data.

[1] http://yann.lecun.com/exdb/mnist/

B. Tested methods

For simplicity, we use the squared loss $L(p, y) = (p - y)^2$ throughout the experiments since the resulting formulation has a closed form solution. The method can be handled by a refined version of Theorem 5.1, which leads to the same conclusions as those of Section VI.

We study the performance of various kernel design methods by changing the spectral coefficients of the initial Gram matrix K, as in Figure 1. Below we write $\bar{\mu}_i$ for the new spectral coefficient of the new Gram matrix $\bar{K}$: i.e., $\bar{K} = \sum_{i=1}^{m} \bar{\mu}_i v_i v_i^T$. We study the following kernel design methods (also see [5]), with a dimension cut off parameter d, so that $\bar{\mu}_i = 0$ when $i > d$.

- $[1, \ldots, 1, 0, \ldots, 0]$: sets the first d coefficients to 1 and the rest to zero:
$$\bar{\mu}_i = \begin{cases} 1 & \text{if } i \leq d \\ 0 & \text{otherwise} \end{cases}$$
This was used in spectral clustering [11].
- Truncated K:
$$\bar{\mu}_i = \begin{cases} \mu_i & \text{if } i \leq d \\ 0 & \text{otherwise} \end{cases}$$
This method is essentially kernel principal component analysis which keeps the d most significant principal components of K.
- K^p: sets the coefficients to the p-th power of the initial coefficients:
$$\bar{\mu}_i = \begin{cases} \mu_i^p & \text{if } i \leq d \\ 0 & \text{otherwise} \end{cases}$$
We set $p = 2, 3, 4$. This accelerates the decay of eigenvalues of K.
- Inverse:
$$\bar{\mu}_i = \begin{cases} 1/(1 - \rho\mu_i/\mu_1) & \text{if } i \leq d \\ 0 & \text{otherwise} \end{cases}$$
ρ is a constant close to 1 (we used 0.999). When the kernel is normalized by $D^{-1/2}WD^{-1/2}$, this is essentially graph Laplacian based semi-supervised learning (e.g., see [17]). However, the unnormalized graph Laplacian regularization, corresponding to unnormalized kernels, is typically defined as $D - W$. It does not correspond to the inverse transformation of W defined here.
Also note that the graph Laplacian formulation sometimes sets $d = m$.
- Y:
$$\bar{\mu}_i = \begin{cases} |Y^T v_i| & \text{if } i \leq d \\ 0 & \text{otherwise} \end{cases}$$
This is the oracle kernel that optimizes our generalization bound. The purpose of testing this oracle method is to validate our analysis by checking whether a good kernel in our theory actually produces good classification performance on real data. Note that in the figures of spectral coefficients we show averaged $\bar{\mu}_i$ over all the classes.

C. Results

Figure 6 shows the spectral coefficients of the above mentioned kernel design methods and the corresponding classification performance on the MNIST data set. The initial kernel is normalized 25-NN, which is defined as $K = \frac{1}{2}D^{-1/2}(D + W)D^{-1/2}$ (see previous section), where for two nodes $i \neq j$, $W_{ij} = 1$ if either the i-th example is one of the 25 nearest neighbors of the j-th example or vice versa; and $W_{ij} = 0$ otherwise. Note that $D + W$ has non-negative eigenvalues. As expected, the results demonstrate that the target spectral coefficients 'Y' decay faster than that of the original kernel K. Therefore it is useful to use kernel design methods that accelerate the eigenvalue decay. The accuracy plot on the right is consistent with our theory. The oracle kernel 'Y' performs well especially when the dimension cut-off is large. With appropriate dimension d, all methods perform better than the supervised base-line (original K) which is below 65%. With appropriate dimension cut-off, all methods perform similarly (over 80%). However, K^p with ($p = 2, 3, 4$) is less sensitive to the cut-off dimension d than the kernel principal component dimension reduction method K. Moreover, the hard threshold method in spectral clustering ($[1, \ldots, 1, 0, \ldots, 0]$) is not stable.

Similar behavior can also be observed with other initial kernels. Figure 7 shows the classification accuracy with the standard Gaussian kernel as the initial kernel K, both with and without normalization, on MNIST. We also used different bandwidth t to illustrate that the behavior of different methods are similar with different t (in a reasonable range). However, we did not try to optimize t. Although for unnormalized kernels, "inverse" is not Laplacian, we include it for completeness. We also observe that "inverse" is more sensitive to the bandwidth t (and more generally, the initial kernel) than other methods. This is because it requires many eigenvalues of the initial kernel to be close to the largest eigenvalue to achieve good eigenvalue decay behavior. Again, we observe that the oracle method performs extremely well. The spectral clustering kernel is sensitive to the cut-off dimension, while K^p with $p = 2, 3, 4$ are quite stable. The standard kernel principal component dimension reduction (method K) performs very well with appropriately chosen dimension cut-off.

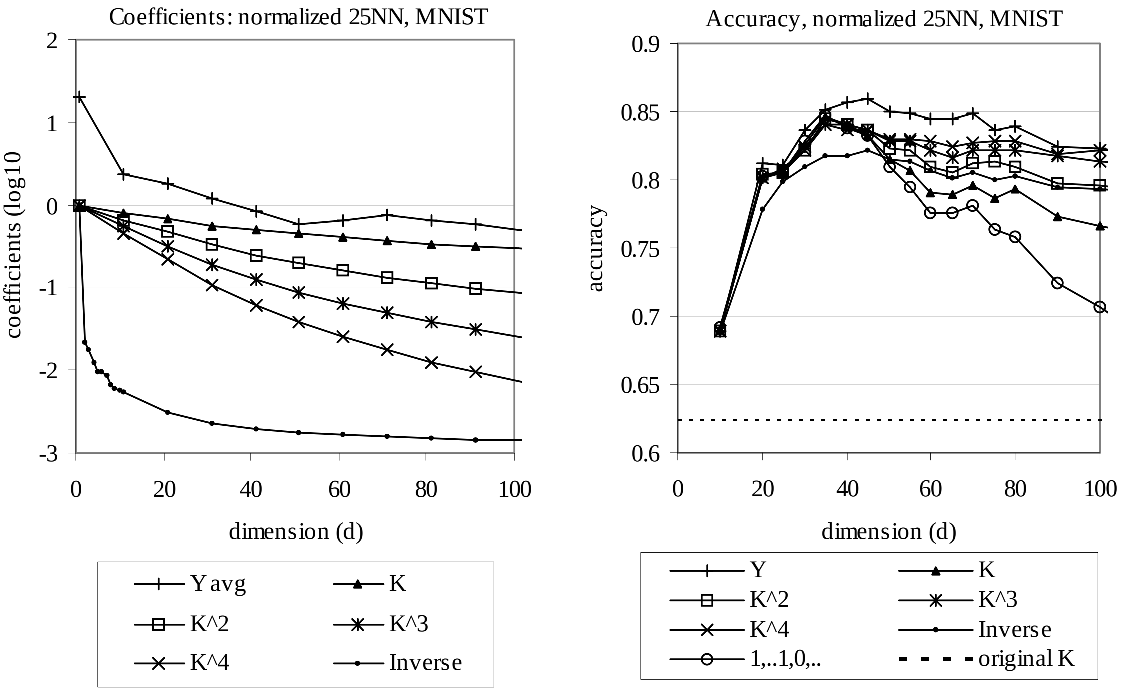

Fig. 6. MNIST. Left: spectral coefficients; right: classification accuracy. Initial kernel is normalized 25-NN kernel.

The unnormalized 25-NN kernel results (Figure 10) also show the similar trend. The experiments are consistent with our theoretical analysis.

Essentially similar results can also be observed on the 20newsgroup data set using various initial kernels. In Figures 11 and 12, the initial kernels are unnormalized and normalized linear kernel, respectively, which are often used for text categorization. The initial kernels in Figure 13 and 14 are unnormalized and normalized 5-NN, respectively.

The right half of Figure 8 plots the spectral coefficients for the input data contaminated by random noise. The noise was generated by randomly choosing and swapping approximately 10% of pixels. Compared with the case without noise (the left half of the same figure), we observe that the random noise 'flattens' the eigenvalue curve (method K) while it does not affect the target coefficients 'Y' much. This is in line with our analysis in Section VI-B. In this case, again, it is useful to use kernel design methods that accelerate the eigenvalue decay. As shown in Figure 9, the noise degrades the performance of the initial K (horizontal line) by 19.1%. The performance of spectral kernel methods is also degraded by noise, but the degradation is as small as 10%.

VIII. Conclusion

We investigated a class of graph-based semi-supervised learning methods. By establishing a graph formulation of kernel learning, we showed that this class of semi-supervised learning methods is equivalent to supervised kernel learning with unsupervised kernel design (explored in [5]). Our formulation is closely related to previously proposed graph learning methods and covers some of them as special cases.

Based on this equivalence formulation, we then studied various theoretical issues for the proposed methods. Firstly, we studied the convergence behavior of the algorithms when the size of unlabeled data set increases. We showed that the methods are well-behaved in the limit under appropriate conditions. We then obtained a generalization bound for graph learning, which we used to analyze the effect of different kernels. In our analysis (which is not necessarily tight), the eigenvalues of the optimal kernel (by minimizing the bound) should decay at the same rate as the target spectral coefficients. In addition, we showed that noise added to the input features can cause the target spectral coefficients to decay faster than the kernel spectral coefficients. Combined with the generalization error bound, our analysis implies that it is helpful to use a kernel with faster spectral decay (than the

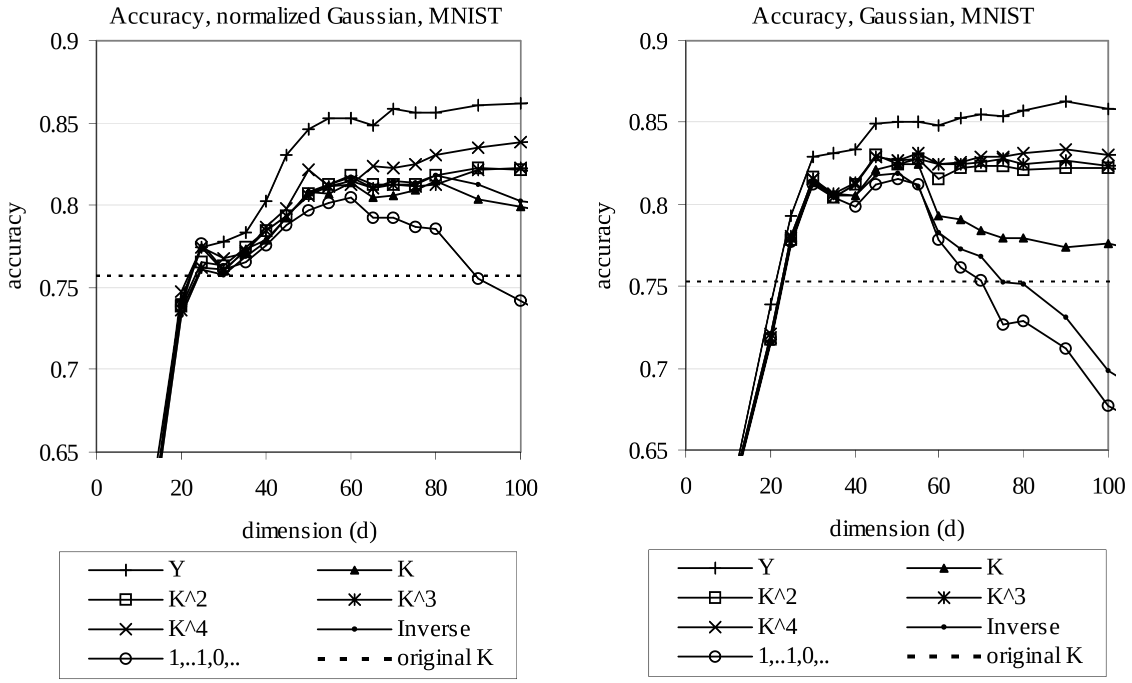

Fig. 7. MNIST. Classification accuracy with Gaussian kernel $k(i,j) = \exp(-||x_i - x_j||_2^2/t)$. Left: normalized Gaussian ($t = 0.1$); right: unnormalized Gaussian ($t = 0.3$).

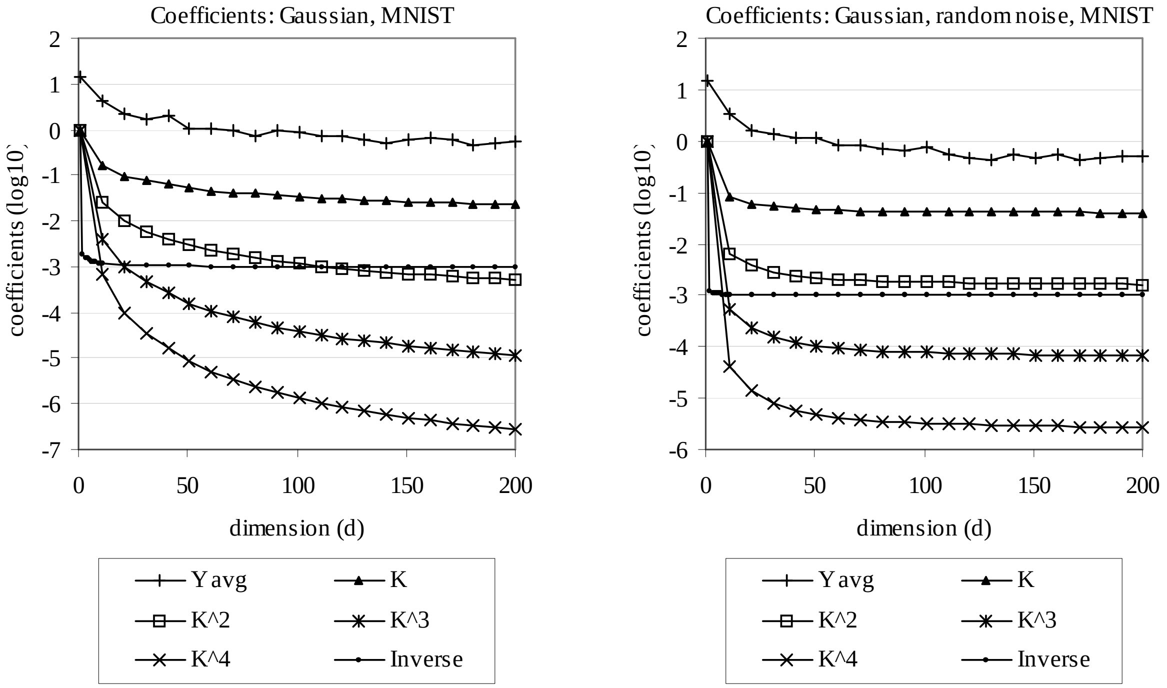

Fig. 8. Noise effect. Spectral coefficients with Gaussian kernel. Left: without noise; right: with noise. MNIST.

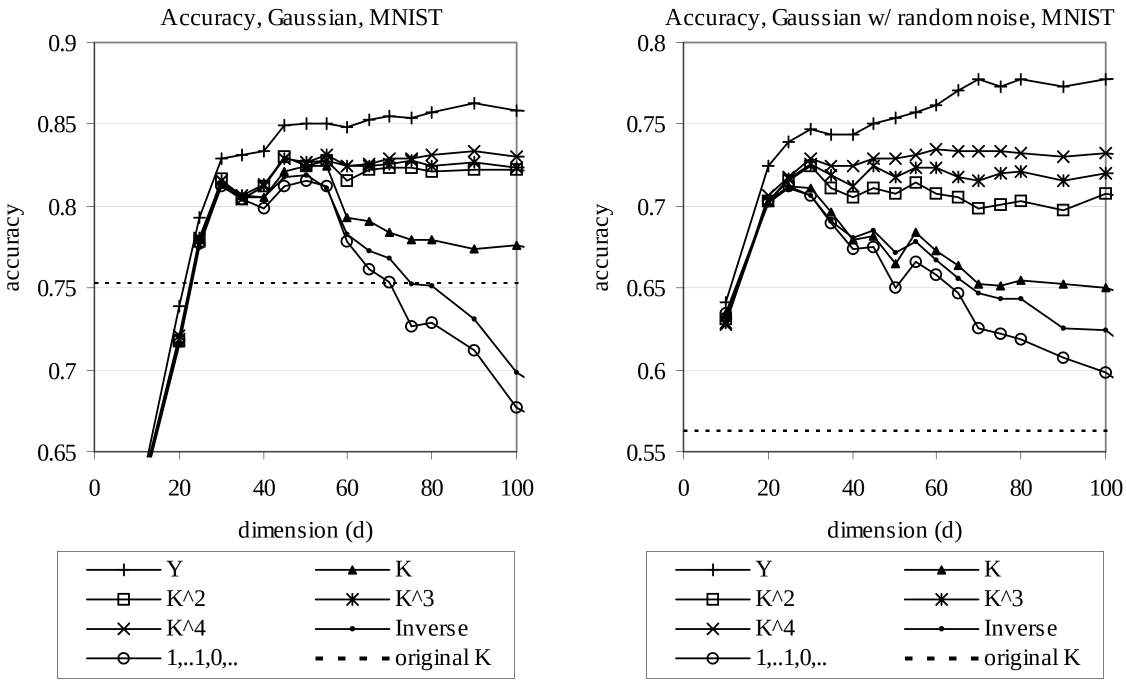

Fig. 9. Noise effect. Classification accuracy with Gaussian kernel. Left: without noise; right: with noise. MNIST.

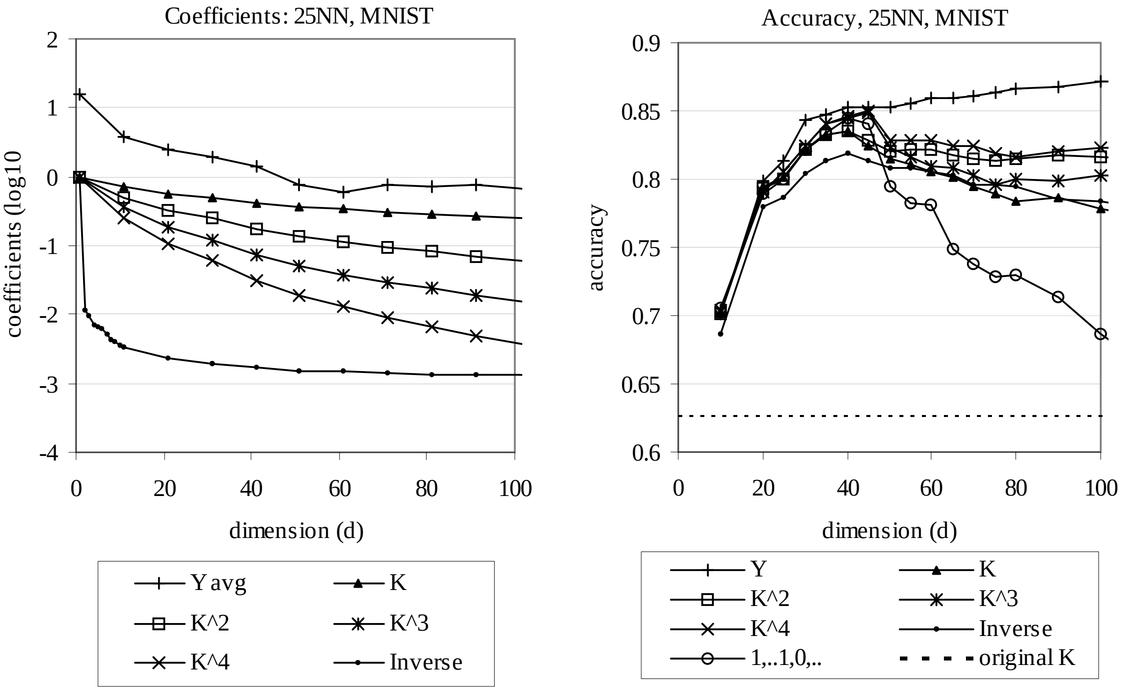

Fig. 10. MNIST. Left: spectral coefficients; right: classification accuracy. Initial kernel is unnormalized 25-NN kernel.

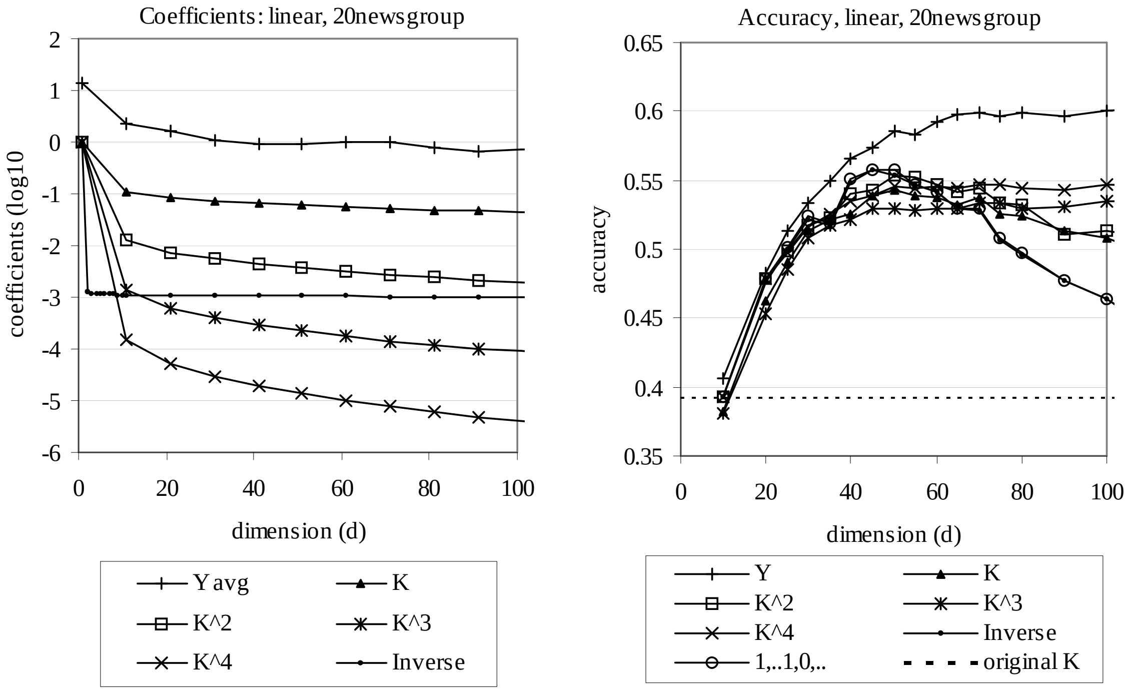

Fig. 11. 20newsgroup. Left: spectral coefficients; right: classification accuracy. Initial kernel is a linear kernel.

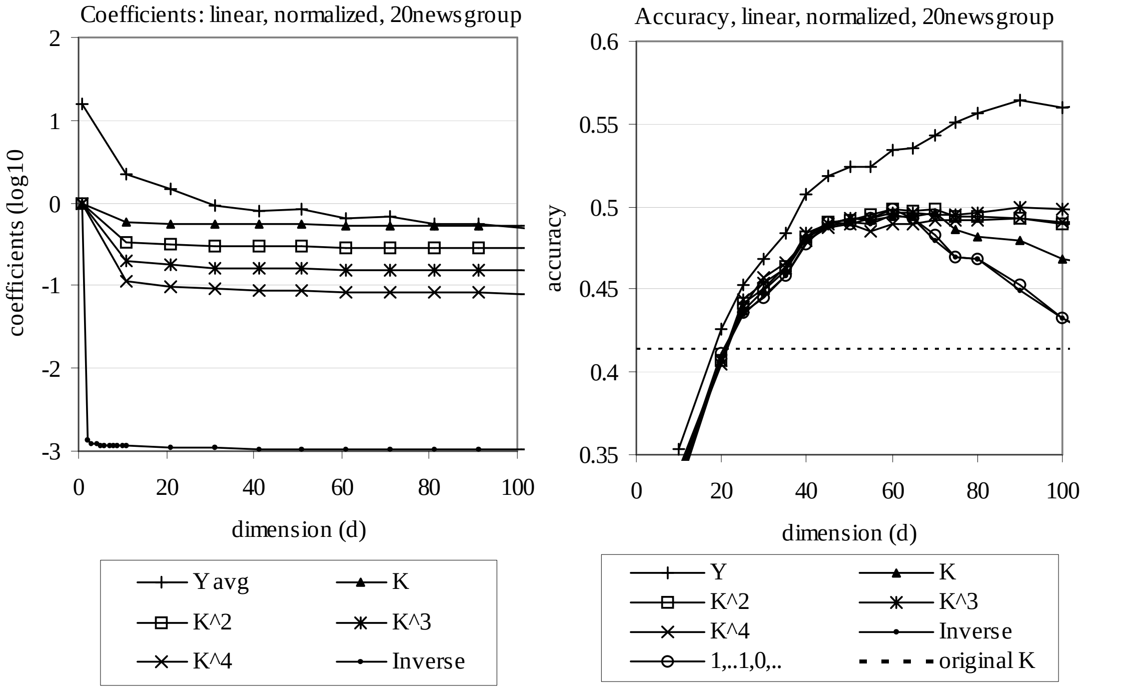

Fig. 12. 20newsgroup. Left: spectral coefficients; right: classification accuracy. Initial kernel is a normalized linear kernel.

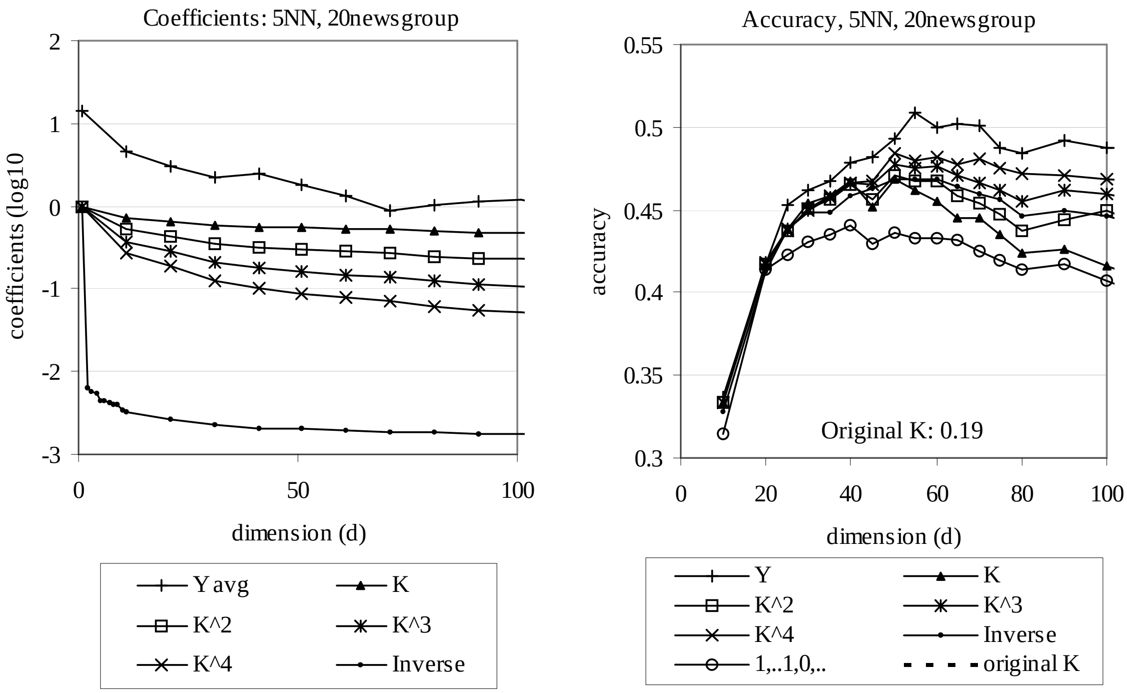

Fig. 13. 20newsgroup. Left: spectral coefficients; right: classification accuracy. Initial kernel is an unnormalized 5-NN kernel.

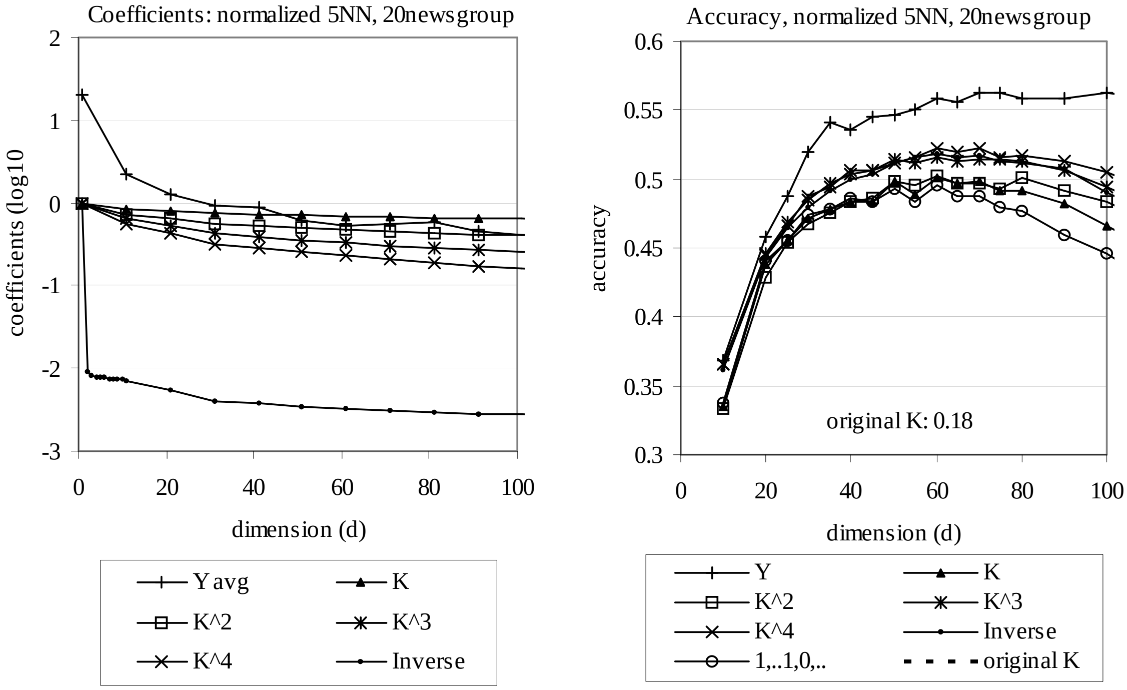

Fig. 14. 20newsgroup. Left: spectral coefficients; right: classification accuracy. Initial kernel is a normalized 5-NN kernel.

initial kernel). In conclusion, our analysis explains why it is often helpful to modify the original kernel eigenvalues to achieve a dimension reduction effect.

Finally we point out that the analysis itself does not provide any data-dependent generalization bound useful for selecting or optimizing kernels (such as what was done in [9]). It only suggests methods to design kernels with fast eigenvalue decay, but does not indicate which method is better. How to derive a tight bound for directly optimizing eigenvalue decay would be an interesting research direction. Another simpler method, which may be employed in practice when the number of labeled data is not too small, is to select from a set of kernels (such as those suggested in the paper) using cross-validation.

APPENDIX

We prove Theorem 5.1. We shall use the following notation: let $i_{n+1} \neq i_1, \ldots, i_n$ be an integer randomly drawn from $\{1, \ldots, m\} - Z_n$, and let $Z_{n+1} = Z_n \cup \{i_{n+1}\}$. Let $\hat{f}(Z_{n+1})$ be the semi-supervised learning method (5) using training data in Z_{n+1}:

$$\hat{f}(Z_{n+1}) = \arg \inf_{f \in R^m} \left[\frac{1}{n} \sum_{i \in Z_{n+1}} L(f_i, Y_i) + \lambda f^T K^{-1} f \right].$$

We employ a stability result from [15], which can be stated in our terminology as: $|\hat{f}_{i_{n+1}}(Z_{n+1}) - \hat{f}_{i_n}(Z_n)| \leq |L_1'(\hat{f}_{i_{n+1}}(Z_{n+1}), Y_{i_{n+1}})| K_{i_{n+1},i_{n+1}}/(2\lambda n)$. This implies that

$$\begin{aligned} & L(\hat{f}_{i_{n+1}}(Z_n), Y_{i_{n+1}}) \\ \leq & L(\hat{f}_{i_{n+1}}(Z_{n+1}), Y_{i_{n+1}}) + K_{i_{n+1},i_{n+1}} \gamma^2/(2\lambda n). \end{aligned}$$

It follows that $\forall f \in R^m$ that is not random:

$$\begin{aligned} & \mathbf{E}_{Z_n} \frac{1}{m-n} \sum_{j \notin Z_n} L(\hat{f}_j(Z_n), Y_j) \\ = & \mathbf{E}_{Z_{n+1}} L(\hat{f}_{i_{n+1}}(Z_n), Y_{i_{n+1}}) \\ \leq & \mathbf{E}_{Z_{n+1}} [L(\hat{f}_{i_{n+1}}(Z_{n+1}), Y_{i_{n+1}}) + K_{i_{n+1},i_{n+1}} \gamma^2/(2\lambda n)] \\ = & \mathbf{E}_{Z_{n+1}} \frac{1}{n+1} \sum_{k \in Z_{n+1}} L(\hat{f}_k(Z_{n+1}), Y_k) + \frac{\mathrm{tr}(K)\gamma^2}{2\lambda n m} \\ \leq & \frac{n}{n+1} \mathbf{E}_{Z_{n+1}} \left[\frac{1}{n} \sum_{k \in Z_{n+1}} L(f_k, Y_k) + \lambda f^T K^{-1} f \right] \\ & \quad + \frac{\mathrm{tr}(K)\gamma^2}{2\lambda n m} \\ = & \left[\frac{1}{m} \sum_{j=1}^{m} L(f_j, Y_j) + \frac{\lambda n}{n+1} f^T K^{-1} f \right] + \frac{\gamma^2 \mathrm{tr}(K)}{2\lambda n m}. \end{aligned}$$

Rie Johnson (formerly, Rie Kubota Ando) Rie Johnson obtained a PhD degree in computer science from Cornell University in 2001. She was a research scientist at IBM T.J. Watson Research Center until 2007. Her research interests are in machine learning and its applications.

Tong Zhang Tong Zhang received a B.A. in mathematics and computer science from Cornell University in 1994 and a Ph.D. in Computer Science from Stanford University in 1998. After graduation, he worked at IBM T.J. Watson Research Center in Yorktown Heights, New York, and Yahoo Research in New York city. He is currently an associate professor of statistics at Rutgers University. His research interests include machine learning, statistical algorithms, their mathematical analysis and applications.

REFERENCES

[1] Y. Altun, M. Belkin, and D. McAllester. Maximum margin semi-supervised learning for structured variables. In *NIPS'05*, 2005.

[2] M. Belkin and P. Niyogi. Semi-supervised learning on Riemannian manifolds. *Machine Learning*, Special Issue on Clustering:209–239, 2004.

[3] M. Belkin and P. Niyogi. Towards a theoretical foundation for Laplacian-based manifold methods. In *COLT'05*, pages 486–500, 2005.

[4] M. Belkin, P. Niyogi, and V. Sindhwani. Manifold regularization: a geometric framework for learning from examples. Technical Report TR-2004-06, Computer Science, University of Chicago, 2004.

[5] O. Chapelle, J. Weston, and B. Schölkopf. Cluster kernels for semi-supervised learning. In *NIPS*, 2003.

[6] E. Gine and V. Koltchinskii. Empirical graph Laplacian approximation of Laplace-Beltrami operators: large sample results. In *The 4th International Conference on High Dimensional Probability*, 2005.

[7] M. Hein. Uniform convergence of adaptive graph-based regularization. In *COLT'06*, 2006.

[8] M. Hein, J.-Y. Audibert, and U. von Luxburg. From graphs to manifolds – weak and strong pointwise consistency of graph Laplacians. In *COLT'05*, pages 470–485, 2005.

[9] G. Lanckriet, N. Cristianini, L. Ghaoui, P. Bartlett, and M. Jordan. Learning the kernel matrix with semidefinite programming. *Journal of Machine Learning Research*, 5:27–72, 2004.

[10] D. J. C. MacKay. Introduction to Gaussian processes. In C. M. Bishop, editor, *Neural Networks and Machine Learning*, NATO ASI Series, pages 133–166. Kluwer, 1998.

[11] A. Y. Ng, M. I. Jordan, and Y. Weiss. On spectral clustering: Analysis and an algorithm. In *NIPS*, pages 849–856, 2001.

[12] R. T. Rockafellar. *Convex analysis*. Princeton University Press, Princeton, NJ, 1970.

[13] M. Szummer and T. Jaakkola. Partially labeled classification with Markov random walks. In *NIPS 2001*, 2002.

[14] G. Wahba. *Spline Models for Observational Data*. CBMS-NSF Regional Conference series in applied mathematics. SIAM, Philadelphia, PA, 1990.

[15] T. Zhang. Leave-one-out bounds for kernel methods. *Neural Computation*, 15:1397–1437, 2003.

[16] T. Zhang. Learning bounds for kernel regression using effective data dimensionality. *Neural Computation*, 17:2077–2098, 2005.

[17] D. Zhou, O. Bousquet, T. Lal, J. Weston, and B. Schölkopf. Learning with local and global consistency. In *NIPS 2003*, pages 321–328, 2004.

[18] X. Zhu, Z. Ghahramani, and J. Lafferty. Semi-supervised learning using Gaussian fields and harmonic functions. In *ICML 2003*, 2003.

[19] X. Zhu, J. Kandola, Z. Ghahramani, and J. Lafferty. Nonparametric transforms of graph kernels for sem-supervised learning. In *NIPS'04*, 2004.

[20] X. Zhu, J. Lafferty, and Z. Ghahramani. Semi-supervised learning: From Gaussian fields to Gaussian processes. Technical Report CMU-CS-03-175, CMU, 2003.