Report No. UIUCDCS-R-2007-2856 **UILU-ENG-2007-1757**

Spectral Regression for Dimensionality Reduction

by

Deng Cai, Xiaofei He, and Jiawei Han

May 2007

Spectral Regression for Dimensionality Reduction*

Deng Cai† Xiaofei He‡ Jiawei Han†

† Department of Computer Science, University of Illinois at Urbana-Champaign

‡ Yahoo! Research Labs

Abstract

Spectral methods have recently emerged as a powerful tool for dimensionality reduction and manifold learning. These methods use information contained in the eigenvectors of a data affinity (*i.e.*, item-item similarity) matrix to reveal low dimensional structure in high dimensional data. The most popular manifold learning algorithms include Locally Linear Embedding, Isomap, and Laplacian Eigenmap. However, these algorithms only provide the embedding results of training samples. There are many extensions of these approaches which try to solve the out-of-sample extension problem by seeking an embedding function in reproducing kernel Hilbert space. However, a disadvantage of all these approaches is that their computations usually involve eigen-decomposition of dense matrices which is expensive in both time and memory. In this paper, we propose a novel dimensionality reduction method, called **Spectral Regression** (SR). SR casts the problem of learning an embedding function into a regression framework, which avoids eigen-decomposition of dense matrices. Also, with the regression based framework, different kinds of regularizers can be naturally incorporated into our algorithm which makes it more flexible. SR can be performed in supervised, unsupervised and semi-supervised situation. It can make efficient use of both labeled and unlabeled points to discover the intrinsic discriminant structure in the data. Experimental results on classification and semi-supervised classification demonstrate the effectiveness and efficiency of our algorithm.

Keywords

Dimensionality Reduction, Spectral Graph Embedding, Regression, Manifold Learning, Discriminant Analysis, Semi-Supervised Learning

1 Introduction

Dimensionality reduction has been a key problem in many fields of information processing, such as machine learning, data mining, information retrieval, and pattern recognition. Practical algorithms in

*The work was supported in part by the U.S. National Science Foundation NSF IIS-05-13678/06-42771 and NSF BDI-05-15813. Any opinions, findings, and conclusions or recommendations expressed here are those of the authors and do not necessarily reflect the views of the funding agencies.

supervised machine learning degrade in performance (prediction accuracy) when faced with many features that are not necessary for predicting the desired output. An important question in the fields of machine learning, knowledge discovery, computer vision and pattern recognition is how to extract a small number of good features. A common way to attempt to resolve this problem is to use dimensionality reduction techniques.

One of the most popular dimensionality reduction algorithms might be Principal Component Analysis (PCA) [20]. PCA performs dimensionality reduction by projecting the original n-dimensional data onto the $d(\ll n)$-dimensional linear subspace spanned by the leading eigenvectors of the data's covariance matrix. Its goal is to find a set of mutually orthogonal basis functions that capture the directions of maximum variance in the data so that the pairwise *Euclidean* distances can be best preserved. If the data is embedded in a linear subspace, PCA is guaranteed to discover the dimensionality of the subspace and produces a compact representation.

In many real world problems, however, there is no evidence that the data is sampled from a linear subspace. For example, it is always believed that the face images are sampled from a nonlinear low-dimensional manifold which is embedded in the high-dimensional ambient space [18]. This motivates us to consider manifold based techniques for dimensionality reduction. Recently, various manifold learning techniques, such as ISOMAP [30], Locally Linear Embedding (LLE) [26] and Laplacian Eigenmap [4] have been proposed which reduce the dimensionality of a *fixed* training set in a way that maximally preserve certain inter-point relationships. LLE and Laplacian Eigenmap are local methods which attempt to preserve local geometry of the data; essentially, they seek to map nearby points on the manifold to nearby points in the low-dimensional representation. ISOMAP is a global method which attempts to preserve geometry at all scales, mapping nearby points on the manifold to nearby points in low-dimensional space, and faraway points to faraway points. One of the major limitations of these methods is that they do not generally provide a functional mapping between the high and low dimensional spaces that are valid both on and off the training data.

There are a lot of approaches that try to address this issue by explicitly requiring an embedding function either linear or in RKHS when minimizing the objective function [17, 6, 33]. They provide natural out-of-sample extensions of Lapalcian Eigenmaps, LLE and Isomap. However, the computation of these methods involves eigen-decomposition of dense matrices which is expensive in both time and memory. It is almost infeasible to apply these approaches on large data sets. Some other approaches address this issue through a kernel view of LLE, Isomap and Laplacian Eigenmaps [5, 13]. They interpret these spectral embedding algorithms as learning the principal eigenfunctions of an operator defined from a kernel and the unknown data generating density. Such kernel is usually data dependant[1]. To obtain the embedding result of an unseen example, we need to calculate the kernel function values of this unseen example with all the training samples which may not be possible in some situations[2].

In this paper, we propose a novel dimensionality reduction algorithm, called *Spectral Regression* (SR).

[1]The kernel function $K(\mathbf{x}_i, \mathbf{x}_j)$ depends not only on $\mathbf{x}_i$ and $\mathbf{x}_j$ but also on the whole data set.

[2]*e.g.*, the data dependant kernel is constructed by integrating label information. To calculate $K(\mathbf{x}_i, \mathbf{x}_j)$, we need to know whether $\mathbf{x}_i$ and $\mathbf{x}_j$ have the same label. Since the label of an unseen example is usually unavailable, we can not calculate the kernel function values of this unseen example with all the training samples.

Table 1: Notations

Notations	Descriptions
m	the number of total training data points
n	the number of features
c	the number of classes
l	the number of labeled data points
l_k	the number of labeled data points in k-th class
$\mathbf{x}_i$	the i-th data point
X	the data matrix
$\mathbf{a}$	the transformation vector (linear projective function)

The proposed algorithm is fundamentally based on regression and spectral graph analysis [7]. It can be performed either in supervised, unsupervised or semi-supervised situation. Specifically, we first construct an affinity graph over both labeled and unlabeled points to discover the intrinsic discriminant structure in the data. This graph is used to learn responses for both labeled and unlabeled points. Once the responses are obtained, the ordinary regression is then applied for learning the embedding function.

The points below highlight several aspects of our approach:

1. SR can be performed in supervised, unsupervised and semi-supervised situation. It can make efficient use of both labeled and unlabeled points to discover the intrinsic discriminant structure in the data. When all the points are labeled, SR provides a Regularized Discriminant Analysis [9] solution. When all the points are unlabeled, SR gives a natural out-of-sample extension for spectral clustering [22, 32] and spectral dimensionality reduction [4].

2. SR uses the regression as the building block, different kinds of regularizers can be naturally incorporated into our algorithm which makes it more flexible.

3. Since the affinity graph is usually sparse and our algorithm is based on the regression framework, the computation can be very fast.

4. Our algorithm may be conducted in the original space or in the reproducing kernel Hilbert space (RKHS) into which data points are mapped. This gives rise to kernel SR.

The paper is structured as follows: in Section 2, we provide a graph embedding view of dimensionality reduction. Our Spectral Regression (SR) algorithm is introduced in Section 3. In Section 4 and 5, we provide a theoretical and computational complexity analysis of our algorithm respectively. The experimental results are presented in Section 6. Finally, we provide some concluding remarks.

2 Graph Embedding View of Dimensionality Reduction

Given m samples $\{\mathbf{x}_i\}_{i=1}^m \subset \mathbb{R}^n$, dimensionality reduction aims at finding $\{\mathbf{z}_i\}_{i=1}^m \subset \mathbb{R}^d, d \ll n$, where $\mathbf{z}_i$ can "represents" $\mathbf{x}_i$. In the past decades, many algorithms, either supervised or unsupervised, have been

proposed to solve this problem. Despite the different motivations of these algorithms, they can be nicely interpreted in a general *graph embedding* framework [6, 18, 33].

Given a graph G with m vertices, each vertex represents a data point. Let W be a symmetric $m \times m$ matrix with W_{ij} having the weight of the edge joining vertices i and j. The G and W can be defined to characterize certain statistical or geometric properties of the data set. The purpose of graph embedding is to represent each vertex of a graph as a low dimensional vector that preserves similarities between the vertex pairs, where similarity is measured by the edge weight.

Let $\mathbf{y} = [y_1, y_2, \cdots, y_m]^T$ be the map from the graph to the real line. The optimal $\mathbf{y}$ tries to minimize

$$\sum_{i,j}(y_i - y_j)^2 W_{ij} \tag{1}$$

under appropriate constraint. This objective function incurs a heavy penalty if neighboring vertices i and j are mapped far apart. Therefore, minimizing it is an attempt to ensure that if vertices i and j are "close" then y_i and y_j are close as well [12]. With some simple algebraic formulations, we have

$$\sum_{i,j}(y_i - y_j)^2 W_{ij} = 2\mathbf{y}^T L \mathbf{y}, \tag{2}$$

where $L = D - W$ is the *graph Laplacian* [7] and D is a diagonal matrix whose entries are column (or row, since W is symmetric) sums of W, $D_{ii} = \sum_j W_{ji}$. Finally, the minimization problem reduces to find

$$\mathbf{y}^* = \arg\min_{\mathbf{y}^T D \mathbf{y} = 1} \mathbf{y}^T L \mathbf{y} = \arg\min \frac{\mathbf{y}^T L \mathbf{y}}{\mathbf{y}^T D \mathbf{y}}. \tag{3}$$

The constraint $\mathbf{y}^T D \mathbf{y} = 1$ removes an arbitrary scaling factor in the embedding. The optimal $\mathbf{y}$'s can be obtained by solving the minimum eigenvalue eigen-problem:

$$L\mathbf{y} = \lambda D \mathbf{y}. \tag{4}$$

Many recently proposed manifold learning algorithms, like Isomap [30], Laplacian Eigenmap [4], Locally Linear Embedding [26], can be interpreted in this framework with different choices of W. Please see [33] for more details.

The graph embedding only presents the mappings for the graph vertices in the training set. For classification purpose, a mapping for all samples, including new test samples, is required. If we choose a linear function, *i.e.*, $y_i = f(\mathbf{x}_i) = \mathbf{a}^T \mathbf{x}_i$, Eq. (3) can be rewritten as:

$$\mathbf{a}^* = \arg\min \frac{\mathbf{y}^T L \mathbf{y}}{\mathbf{y}^T D \mathbf{y}} = \arg\min \frac{\mathbf{a}^T X L X^T \mathbf{a}}{\mathbf{a}^T X D X^T \mathbf{a}}, \tag{5}$$

where $X = [\mathbf{x}_1, \cdots, \mathbf{x}_m]$. The optimal $\mathbf{a}$'s are the eigenvectors corresponding to the minimum eigenvalue of eigen-problem:

$$XLX^T \mathbf{a} = \lambda X D X^T \mathbf{a}. \tag{6}$$

If we choose a function in RKHS, *i.e.*, $y_i = f(\mathbf{x}_i) = \sum_{j=1}^m \alpha_j K(\mathbf{x}_j, \mathbf{x}_i)$, $K(\mathbf{x}_j, \mathbf{x}_i)$ is the Mercer kernel of RKHS $\mathcal{H}_K$. Eq. (3) can be rewritten as:

$$\boldsymbol{\alpha}^* = \arg\min \frac{\mathbf{y}^T L \mathbf{y}}{\mathbf{y}^T D \mathbf{y}} = \arg\min \frac{\boldsymbol{\alpha}^T K L K \boldsymbol{\alpha}}{\boldsymbol{\alpha}^T K D K \boldsymbol{\alpha}}, \tag{7}$$

where K is $m \times m$ gram matrix $K_{ij} = K(\mathbf{x}_i, \mathbf{x}_j)$ and $\boldsymbol{\alpha} = [\alpha_1, \cdots, \alpha_m]^T$. The optimal $\boldsymbol{\alpha}$'s are the eigenvectors corresponding to the minimum eigenvalue of eigen-problem:

$$KLK\boldsymbol{\alpha} = \lambda KDK\boldsymbol{\alpha}. \tag{8}$$

With different choices of W, this framework leads to many popular linear dimensionality reduction algorithms, *e.g.*, Linear Discriminant Analysis (LDA) [10], Locality Preserving Projections (LPP) [17], Neighborhood Preserving Embedding (NPE) [16] and their kernel extensions, *e.g.*, Kernel Discriminant Analysis (KDA) [2], Kernel LPP [17], Kernel Eigenmaps [6].

Although the affinity matrix W could be sparse, the matrices XLX^T, XDX^T, KLK and KDK are all dense. Therefor, solving the eigen-problem in Eqn (6) and Eqn (8) are both time and memory consuming. Moreover, to get a stable solution of eigen-problem in Eqn (6), the matrices XDX^T is required to be non-singular [11] which is not true when the number of features is larger than the number of samples. In this case, some additional preprocessing steps (*e.g.*, SVD) are thus needed.

Suppose $rank(X) = r$, the SVD decomposition of X is

$$X = U\Sigma V^T \tag{9}$$

where $\Sigma = diag(\sigma_1, \cdots, \sigma_r)$ and $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_r > 0$ are the singular values of X, $U = [\mathbf{u}_1, \cdots, \mathbf{u}_r] \in \mathbb{R}^{n \times r}$ and $\mathbf{u}_i$'s are called left singular vectors, $V = [\mathbf{v}_1, \cdots, \mathbf{v}_r] \in \mathbb{R}^{m \times r}$ and $\mathbf{v}_i$'s are called right singular vectors. Let $\widetilde{X} = U^T X = \Sigma V^T$ and $\mathbf{b} = U^T \mathbf{a}$, we have

$$\mathbf{a}^T XLX^T \mathbf{a} = \mathbf{a}^T U\Sigma V^T LV\Sigma U^T \mathbf{a} = \mathbf{b}^T \widetilde{X} L \widetilde{X}^T \mathbf{b}$$

and

$$\mathbf{a}^T XDX^T \mathbf{a} = \mathbf{a}^T U\Sigma V^T DV\Sigma U^T \mathbf{a} = \mathbf{b}^T \widetilde{X} D \widetilde{X}^T \mathbf{b}$$

Now, the objective function of linear graph embedding in (5) can be rewritten as:

$$\mathbf{b}^* = \arg\min \frac{\mathbf{b}^T \widetilde{X} L \widetilde{X}^T \mathbf{b}}{\mathbf{b}^T \widetilde{X} D \widetilde{X}^T \mathbf{b}},$$

and the optimal $\mathbf{b}$'s are the eigenvectors corresponding to the minimum eigenvalue of eigen-problem:

$$\widetilde{X} L \widetilde{X}^T \mathbf{b} = \lambda \widetilde{X} D \widetilde{X}^T \mathbf{b}. \tag{10}$$

It is clear that $\widetilde{X} D \widetilde{X}^T$ is nonsingular and the above eigen-problem can be stably solved. After we get $\mathbf{b}^*$, the $\mathbf{a}^*$ can be obtained by solving a set of linear equations systems $U^T \mathbf{a} = \mathbf{b}^*$. By noticing that given U, $\mathbf{b}^*$, there will be infinitely many solutions of $\mathbf{a}$ which satisfy this equations system[3]. Among all these solutions, $\mathbf{a}^* = U\mathbf{b}^*$ is obviously one of them. The similar approach can also be used to handle the singularity of KDK [2].

[3]Unless $n < m$ and $rank(X) = n$. In this case, U will be an orthogonal matrix and there is an unique solution of equation $U^T \mathbf{a} = \mathbf{b}^*$ which is exactly $U\mathbf{b}^*$.

The SVD step further increases the computational cost. In some applications (*e.g.*, text processing), the data matrix is sparse which can be fit into the memory even with a large number of both samples and features. However, the singular vector matrices of SVD decomposition on data matrix are dense, thus may not be able to be fit into the memory. In this case, all these linear graph embedding approaches can not be applied.

Since we have $L = D - W$, it is easy to check that the **minimization** problem in Eqn. (3) is equivalent to the following **maximization** problem:

$$\mathbf{y}^* = \arg\max_{\mathbf{y}^T D\mathbf{y}=1} \mathbf{y}^T W\mathbf{y} = \arg\max \frac{\mathbf{y}^T W\mathbf{y}}{\mathbf{y}^T D\mathbf{y}}.$$

and the optimal $\mathbf{y}$'s are the **maximum** eigenvectors of eigen-problem

$$W\mathbf{y} = \lambda D\mathbf{y}. \tag{11}$$

The eigen-problems of obtaining linear and kernel embedding functions can also be rewritten as

$$XWX^T\mathbf{a} = \lambda XDX^T\mathbf{a}. \tag{12}$$

and

$$KWK\boldsymbol{\alpha} = \lambda KDK\boldsymbol{\alpha}. \tag{13}$$

respectively. This maximum eigen-problem formulation in some cases can provide a more numerically stable solution [11]. In the following of our paper, we will develop our algorithm based on this formulation.

3 The Spectral Regression Algorithm

Clearly, seeking a linear function (or a function in RKHS) which minimizes the objective function is a natural extension of the spectral dimensionality reduction algorithms. However, the high computational cost in both time and memory restricts these approaches to be applied on large scale data sets. In this section, we describe our algorithm which can solve this problem.

In order to solve the this eigen-problem in Eqn. (12) and (13) efficiently, we use the following theorem:

Theorem 1 *Let $\boldsymbol{y}$ be the eigenvector of eigen-problem in Eqn. (11) with eigenvalue λ. If $X^T\boldsymbol{a} = \boldsymbol{y}$, then $\boldsymbol{a}$ is the eigenvector of eigen-problem in Eqn. (12) with the same eigenvalue λ. If $K\boldsymbol{\alpha} = \boldsymbol{y}$, then $\boldsymbol{\alpha}$ is the eigenvector of eigen-problem in Eqn. (13) with the same eigenvalue λ.*

Proof We have $W\mathbf{y} = \lambda D\mathbf{y}$. At the left side of Eqn. (12), replace $X^T\mathbf{a}$ by $\mathbf{y}$, we have

$$XWX^T\mathbf{a} = XW\mathbf{y} = X\lambda D\mathbf{y} = \lambda XD\mathbf{y} = \lambda XDX^T\mathbf{a}$$

Thus, $\mathbf{a}$ is the eigenvector of eigen-problem Eqn. (12) with the same eigenvalue λ. The second part of the theorem can be proved by the same approach.

Theorem (1) shows that instead of solving the eigen-problem Eqn. (12), the linear embedding functions can be acquired through two steps:

1. Solve the eigen-problem in Eqn. (11) to get $\mathbf{y}$.

2. Find $\mathbf{a}$ which satisfies $X^T\mathbf{a} = \mathbf{y}$. In reality, such $\mathbf{a}$ might not exist. A possible way is to find $\mathbf{a}$ which can best fit the equation in the least squares sense:

$$\mathbf{a} = \arg\min_{\mathbf{a}} \sum_{i=1}^{m} (\mathbf{a}^T\mathbf{x}_i - y_i)^2 \tag{14}$$

   where y_i is the i-th element of $\mathbf{y}$.

The advantages of this two-step approach are as follows:

1. The matrix D is guaranteed to be positive definite and therefor the eigen-problem in Eqn. (11) can be stably solved. Moreover, both L and D are sparse matrices. The top c eigenvectors of eigen-problem in Eqn. (11) can be efficiently calculated with Lanczos algorithms [11].

2. The technique to solve the least square problem is already matured [11] and there exist many efficient iterative algorithms (*e.g.*, LSQR [24]) that can handle very large scale least square problems.

In the situation that the number of samples is smaller than the number of features, the minimization problem (14) is *ill posed*. We may have infinite many solutions for the linear equations system $X^T\mathbf{a} = \mathbf{y}$ (the system is underdetermined). The most popular way to solve this problem is to impose a penalty on the norm of $\mathbf{a}$:

$$\mathbf{a} = \arg\min_{\mathbf{a}} \left(\sum_{i=1}^{m} \left(\mathbf{a}^T\mathbf{x}_i - y_i\right)^2 + \alpha\|\mathbf{a}\|^2 \right) \tag{15}$$

This is so called regularization and is well studied in statistics. The regularized least square is also called ridge regression [15]. The $\alpha \geq 0$ is a parameter to control the amounts of shrinkage. Now we can see the third advantage of the two-step approach:

3 Since the regression was used as a building block, the regularization techniques can be easily incorporated and produce more stable and meaningful solutions, especially when there exist a large amount of features [15].

If the linear regression is replaced by the kernel regression, we can find the embedding functions in the RKHS. The analysis is similar to the above and is omitted to avoid unnecessary repetition.

Since our approach essentially performs regression after the spectral analysis of the graph, we called it *Spectral Regression* (SR). Given a labeled set $\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_l$ and an unlabeled set $\mathbf{x}_{l+1}, \mathbf{x}_{l+2}, \cdots, \mathbf{x}_m$ in $\mathbf{R}^n$. These samples belong to c classes and let l_k be the number of labeled samples in the k-th class ($\sum_{k=1}^{c} l_k = l$). The algorithmic procedure of SR is stated below:

1. **Constructing the adjacency graph**: Let G denote a graph with m nodes. The i-th node corresponds to the sample $\mathbf{x}_i$. We construct the graph G through the following three steps to model the local structure as well as the label information:

(a) Put an edge between nodes i and j if $\mathbf{x}_i$ is among p nearest neighbors of $\mathbf{x}_j$ or $\mathbf{x}_j$ is among p nearest neighbors of $\mathbf{x}_i$.

(b) Put an edge between nodes i and j if $\mathbf{x}_i$ shares the same label with $\mathbf{x}_j$.

(c) Remove the edge between nodes i and j if the label of $\mathbf{x}_i$ is different from that of $\mathbf{x}_j$.

2. **Choosing the weights**: W is a sparse symmetric $m \times m$ matrix with W_{ij} having the weight of the edge joining vertices i and j.

   (a) If there is no edge between i and j, $W_{ij} = 0$.

   (b) Otherwise,

$$W_{ij} = \begin{cases} 1/l_k, & \text{if } \mathbf{x}_i \text{ and } \mathbf{x}_j \text{ both belong} \\ & \text{to the } k\text{-th class;} \\ \delta \cdot s(i,j), & \text{otherwise.} \end{cases} \tag{16}$$

   where $0 < \delta \leq 1$ is the parameter to adjust the weight between supervised information and unsupervised neighbor information. $s(i,j)$ is a function to evaluate the similarity between $\mathbf{x}_i$ and $\mathbf{x}_j$ and we have two variations.

   (a) Heat kernel [parameter $\sigma \in \mathbb{R}$]:

$$s(i,j) = e^{-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma^2}}$$

   (b) Simple-minded [no parameter]:

$$s(i,j) = 1$$

3. **Responses generation**: Find $\mathbf{y}_0, \mathbf{y}_1, \cdots, \mathbf{y}_{c-1}$, the largest c generalized eigenvectors of eigenproblem:

$$W\mathbf{y} = \lambda D\mathbf{y} \tag{17}$$

   where D is a diagonal matrix whose (i,i)-*th* element equals to the sum of the i-th column (or row, since W is symmetric) of W. It is straightforward to show that the first eigenvector is a vector of all ones with eigenvalue 1 [7].

4. **Regularized least squares**: Find $c-1$ vectors $\mathbf{a}_1, \cdots, \mathbf{a}_{c-1} \in \mathbb{R}^n$. $\mathbf{a}_k$ $(k = 1, \cdots, c-1)$ is the solution of regularized least square problem:

$$\mathbf{a}_k = \arg\min_{\mathbf{a}} \left(\sum_{i=1}^{m} (\mathbf{a}^T \mathbf{x}_i - y_i^k)^2 + \alpha \|\mathbf{a}\|^2 \right) \tag{18}$$

   where y_i^k is the i-th element of $\mathbf{y}_k$. It is easy to check that $\mathbf{a}_k$ is the solution of the linear equations system:

$$(XX^T + \alpha I)\mathbf{a}_k = X\mathbf{y}_k \tag{19}$$

   where I is a $n \times n$ identity matrix. The canonical Gaussian elimination method can be used to solve this linear equations system [11]. When X is large, some efficient iterative algorithms (*e.g.*, LSQR [24]) can be used to directly solve the above regularized least square problem in Eqn. (18).

5. **SR Embedding**: Let $A = [\mathbf{a}_1, \mathbf{a}_2, \cdots, \mathbf{a}_{c-1}]$, A is a $n \times (c-1)$ transformation matrix. The samples can be embedded into $c-1$ dimensional subspace by

$$\mathbf{x} \rightarrow \mathbf{z} = A^T \mathbf{x} \tag{20}$$

If embedding functions in RKHS are required, the 4th step can be replaced by regularized kernel least squares as follows:

4′ **Regularized kernel least squares**: Find $c-1$ vectors $\boldsymbol{\alpha}_1, \cdots, \boldsymbol{\alpha}_{c-1} \in \mathbb{R}^m$. $\boldsymbol{\alpha}_k$ $(k = 1, \cdots, c-1)$ is the solution the linear equations system:

$$(K + \alpha I)\boldsymbol{\alpha}_k = \mathbf{y}_k \tag{21}$$

where K is $m \times m$ gram matrix $K_{ij} = K(\mathbf{x}_i, \mathbf{x}_j)$ and $K(\mathbf{x}_i, \mathbf{x}_j)$ is the Mercer kernel of RKHS $\mathcal{H}_K$. It can be easily verified that function $f(\mathbf{x}) = \sum_{i=1}^m \alpha_i^k K(\mathbf{x}, \mathbf{x}_i)$ is the solution of the following regularized kernel least square problem [27]:

$$\min_{f \in \mathcal{H}_K} \sum_{i=1}^m \left(f(\mathbf{x}_i) - y_i^k\right)^2 + \alpha \|f\|_K^2 \tag{22}$$

where α_i^k is the i-th element of vector $\boldsymbol{\alpha}_k$.

5′ **KSR Embedding**: Let $\Theta = [\boldsymbol{\alpha}_1, \cdots, \boldsymbol{\alpha}_{c-1}]$, Θ is a $m \times (c-1)$ transformation matrix. The samples can be embedded into $c-1$ dimensional subspace by

$$\mathbf{x} \rightarrow \mathbf{z} = \Theta^T K(:, \mathbf{x}) \tag{23}$$

where $K(:, \mathbf{x}) \doteq [K(\mathbf{x}_1, \mathbf{x}), \cdots, K(\mathbf{x}_m, \mathbf{x})]^T$

4 Theoretical Analysis

SR (KSR) uses regularized least squares to get the embedding functions. When $\alpha > 0$, the regularized solution will not satisfy the linear equations system $X^T\mathbf{a} = \mathbf{y}$ and $\mathbf{a}$ will not be the eigenvector of of eigen-problem in Eqn. (12) (So as the KSR solution $\boldsymbol{\alpha}$). It is interesting and important to see when SR gives the exact solutions of eigen-problem (12) and (13). Specifically, we have the following theorem:

Theorem 2 *Suppose $\boldsymbol{y}$ is the eigenvector of eigen-problem in Eqn. (11), if $\boldsymbol{y}$ is in the space spanned by row vectors of X, the corresponding projective function $\boldsymbol{a}$ calculated in SR will be the eigenvector of eigen-problem in Eqn. (12) as α deceases to zero. If $\boldsymbol{y}$ is in the space spanned by row (or column, since K is symmetric) vectors of K, the corresponding projective function $\boldsymbol{\alpha}$ calculated in KSR will be the eigenvector of eigen-problem in Eqn. (13) as α deceases to zero.*

Proof Suppose $rank(X) = r$, the SVD decomposition of X is

$$X = U\Sigma V^T \tag{24}$$

where $\Sigma = diag(\sigma_1, \cdots, \sigma_r)$, $U \in \mathbb{R}^{n\times r}$, $V \in \mathbb{R}^{m\times r}$ and we have $U^TU = V^TV = I$. The $\mathbf{y}$ is in the space spanned by row vectors of X, therefor, $\mathbf{y}$ is in the space spanned by column vectors of V. Thus, $\mathbf{y}$ can be represented as the linear combination of the column vectors of V. Moreover, the combination is unique because the column vectors of V are linear independent. Suppose the combination coefficients are $b_1, \cdots, b_r$. Let $\mathbf{b} = [b_1, \cdots, b_r]^T$, we have:

$$\begin{aligned} V\mathbf{b} = \mathbf{y} \quad &\Rightarrow \quad V^TV\mathbf{b} = V^T\mathbf{y} \\ &\Rightarrow \quad \mathbf{b} = V^T\mathbf{y} \\ &\Rightarrow \quad VV^T\mathbf{y} = \mathbf{y} \end{aligned} \tag{25}$$

To continue our proof, we need introduce the concept of pseudo inverse of a matrix [25], which we denote as $(\cdot)^+$. Specifically, pseudo inverse of the matrix X can be computed by the following two ways:

$$X^+ = V\Sigma^{-1}U^T$$

and

$$X^+ = \lim_{\lambda\to 0}(X^TX + \lambda I)^{-1}X^T$$

The above limit exists even if X^TX is singular and $(X^TX)^{-1}$ does not exist [25]. Thus, the regularized least squares solution in SR

$$\begin{aligned} \mathbf{a} &= \left(XX^T + \alpha I\right)^{-1}X\mathbf{y} \\ &= (X^T)^+\mathbf{y} \qquad (\alpha \to 0) \\ &= U\Sigma^{-1}V^T\bar{\mathbf{y}} \end{aligned}$$

Combine with the equation in Eqn. (25), we have

$$\begin{aligned} X^T\mathbf{a} &= V\Sigma U^T\mathbf{a} \\ &= V\Sigma U^TU\Sigma^{-1}V^T\mathbf{y} = VV^T\mathbf{y} = \mathbf{y} \end{aligned}$$

By Theorem (1), $\mathbf{a}$ is the eigenvector of eigen-problem in Eqn. (12). The KSR part of the theorem can be proved with the same approach.

When the the number of features is larger than the number of samples, the sample vectors are usually linearly independent, *i.e.*, $rank(X) = m$. In this case, we will have a stronger conclusion for SR which is shown in the following Corollary.

Corollary 3 *If the sample vectors are linearly independent,* i.e., $rank(X) = m$, *all the projective functions in SR are the eigenvectors of eigen-problem in Eqn. (12) as α deceases to zero. These solutions are identical to the linear graph embedding solutions as described in Section (2).*

Proof Since $rank(X) = m$, all the m eigenvectors $\mathbf{y}_k$ of eigen-problem (11) are in the space spanned by row vectors of X. By Theorem (2), all m corresponding $\mathbf{a}_k$ of SR are eigenvectors of eigen-problem in Eqn. (12) as α decreases to zero. They are

$$\mathbf{a}_k^{SR} = U\Sigma^{-1}V^T\mathbf{y}_k.$$

Consider the eigen-problem in Eqn. (10), since the m eigenvectors $\mathbf{y}_k$ are also in the space spanned by row vectors of $\widetilde{X} = U^TX = \Sigma V^T$, eigenvector $\mathbf{b}_k$ will be the solution of linear equations system $\widetilde{X}^T\mathbf{b}_k = \mathbf{y}_k$. The row vectors of $\widetilde{X} = \Sigma V^T$ are linearly independent, thus $\mathbf{b}_k$ is unique and

$$\mathbf{b}_k = \Sigma^{-1}V^T\mathbf{y}_k.$$

Thus, the projective functions of Linear Graph Embedding (LDE) described in section 2

$$\mathbf{a}_k^{LDE} = U\mathbf{b}_k = U\Sigma^{-1}V^T\mathbf{y}_k = \mathbf{a}_k^{SR}$$

For KSR, if the Gaussian RBF kernel is used, a similar conclusion will hold only requires that all the sample vectors are different.

Corollary 4 *If all the sample vectors are different, all projective functions in KSR with Gaussian RBF kernel are eigenvectors of eigen-problem in Eqn. (13) when* $\alpha = 0$.

Proof When m samples are distinct points, the Gaussian RBF gram matrix K is of full rank [21, 27]. The linear equations system $K\boldsymbol{\alpha} = \mathbf{y}$ has unique solution $\boldsymbol{\alpha} = K^{-1}\mathbf{y}$ which is the eigenvector of eigen-problem in Eqn. (13).

The corollary (3) also implies that when the sample vectors are linear independent, which will usually be the case when the number of features is larger than the number of samples, the embedding functions of linear SR will give the exactly same embedding results on training set as those spectral embedding algorithms (*e.g.*, Laplacian Eigenmaps) if the same W is used as α decreases to zero. For kernel SR, if the Gaussian RBF kernel is used, corollary (4) shows that the same conclusion will hold if all the sample vectors are different. While the embedding functions in linear SR and kernel SR are defined everywhere, both on and off training samples.

4.1 Connection to LDA in Supervised Setting

With all the data points labeled, we have $\sum_{k=1}^{c} l_k = l = m$. Without loss of generality, we assume that the data points in $\{\mathbf{x}_1, \cdots, \mathbf{x}_m\}$ are ordered according to their labels. Let $W^{(k)}$ be a $l_k \times l_k$ matrix with all elements being $1/l_k$. We can easily see that the matrix W in Eqn. (16) has a block-diagonal structure

$$W = \begin{bmatrix} W^{(1)} & 0 & \cdots & 0 \\ 0 & W^{(2)} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & W^{(c)} \end{bmatrix} \tag{26}$$

We have $D = I$ and the generalized eigen-problem in Eq. (17) reduces to an ordinary eigen-problem

$$W\mathbf{y} = \lambda\mathbf{y}. \tag{27}$$

With Theorem (2), we know that SR essentially provides the solutions of eigen-problem

$$XWX^T\mathbf{a} = \lambda XX^T\mathbf{a}. \tag{28}$$

Let $X^{(k)} = [\mathbf{x}_1^{(k)}, \cdots, \mathbf{x}_{l_k}^{(k)}]$ denote the data matrix of k-th class, we have

$$\begin{aligned} XWX^T =& \sum_{k=1}^{c} X^{(k)}W^{(k)}(X^{(k)})^T \\ =& \sum_{k=1}^{c} \frac{1}{l_k}\left(\sum_{i=1}^{l_k}\mathbf{x}_i^{(k)}\sum_{i=1}^{l_k}(\mathbf{x}_i^{(k)})^T\right) \\ =& \sum_{k=1}^{c} l_k\ \boldsymbol{\mu}^{(k)}(\boldsymbol{\mu}^{(k)})^T \end{aligned} \tag{29}$$

where $\boldsymbol{\mu}^{(k)} = \frac{1}{l_k}\sum_{i=1}^{l_k}\mathbf{x}_i^{(k)}$ is the mean vector of the k-th class. If the data are centered, *i.e.*, the total mean vector $\boldsymbol{\mu} = \frac{1}{m}\sum_{i=1}^{m}\mathbf{x}_i = \mathbf{0}$, we have

$$XWX^T = \sum_{k=1}^{c} l_k(\boldsymbol{\mu}^{(k)} - \boldsymbol{\mu})(\boldsymbol{\mu}^{(k)} - \boldsymbol{\mu})^T = S_b$$

and

$$XX^T = \sum_{i=1}^{m}\mathbf{x}_i\mathbf{x}_i^T = \sum_{i=1}^{m}(\mathbf{x}_i - \boldsymbol{\mu})(\mathbf{x}_i - \boldsymbol{\mu})^T = S_t$$

S_b is called the between-class scatter matrix and S_t is the total scatter matrix [10]. The eigen-problem in Eq. (28) can also be written as

$$S_b\mathbf{a} = \lambda S_t\mathbf{a}. \tag{30}$$

Since $S_t = S_b + S_w$ where S_w is the within-class scatter matrix, the maximum eigenvalue problem (30) is equivalent to

$$S_b\mathbf{a} = \lambda S_w\mathbf{a}. \tag{31}$$

The latter is exactly the eigen-problem of Linear Discriminant Analysis (LDA) [10]. Thus, with centered data, SR in supervised setting provides the LDA solution.

LDA is a canonical approach for classification and dimensionality reduction . It seeks directions on which the data points of different classes are far away from each other while requiring data points of the same class to be close to each other. The classification is based on the distances of the test point to the class centroids in the reduced subspace. It is guaranteed to be Bayesian optimal when each class is Gaussian with the same covariance matrix [10, 15].

Now let us examine the eigenvector $\mathbf{y}$ in Eq. (27). The W is block-diagonal, thus, its eigenvalues and eigenvectors are the union of the eigenvalues and eigenvectors of its blocks (the latter padded appropriately with zeros). It is straightforward to show that $W^{(k)}$ has eigenvector $\mathbf{e}^{(k)} \in \mathbb{R}^{l_k}$ associated

with eigenvalue 1, where $\mathbf{e}^{(k)} = [1, 1, \cdots, 1]^T$. Also there is only one non-zero eigenvalue of $W^{(k)}$ because the rank of $W^{(k)}$ is 1. Thus, there are exactly c eigenvectors of W with the same eigenvalue 1. These eigenvectors are

$$\mathbf{y}_k = [\underbrace{0, \cdots, 0}_{\sum_{i=1}^{k-1} l_i}, \underbrace{1, \cdots, 1}_{l_k}, \underbrace{0, \cdots, 0}_{\sum_{i=k+1}^{c} l_i}]^T \tag{32}$$

Since 1 is a repeated eigenvalue of W, we could just pick any other c orthogonal vectors in the space spanned by $\{\mathbf{y}_k\}$, and define them to be our first c eigenvectors. The vector of all ones is naturally in the spanned space. This vector is useless since the responses of all the data points are the same. In reality, we can pick the vector of all ones as our first eigenvector and use Gram-Schmidt process to get the remaining $c-1$ orthogonal eigenvectors. The vector of all ones can then be removed. It is important to note that the values of the i-th and j-th positions of any vector $\mathbf{y}$ in the space spanned by $\{\mathbf{y}_k\}$ will be the same as long as $\mathbf{x}_i$ and $\mathbf{x}_j$ belong to the same class.

Besides the advantages of SR we described in the previous section, two points should be emphasized for using SR instead of directly applying LDA:

1. LDA needs to solve the eigen-problem in Eq. (31) which could be very computationally expensive for high dimensional data in large scale problem. The eigen-decomposition step of SR in Eqn. (17) is trivial with the W constructed as in Eqn. (26). Thus, SR only needs to solve $c-1$ regularized least squares problems which can be very efficient.

2. When the parameter $\alpha > 0$ in the fourth step, SR gives a regularized LDA solution [9]. It was showed that regularized LDA can get better performance than ordinary LDA, especially we have a large amount of features [14].

4.2 Connections to Laplacian Eigenmap and LPP in Unsupervised Setting

When all the data points are unlabeled, W in the second step of our algorithm is simply a p-nearest neighbor graph. Let $L = D - W$, it is easy to check that the largest eigenvectors of the eigen-problem (17) are equivalent to the smallest eigenvectors of $L\mathbf{y} = \lambda' D\mathbf{y}$. Specifically, we have

$$\begin{aligned} W\mathbf{y} = \lambda D\mathbf{y} &\Rightarrow (D - L)\mathbf{y} = \lambda D\mathbf{y} \\ &\Rightarrow L\mathbf{y} = (1 - \lambda)D\mathbf{y} = \lambda' D\mathbf{y} \end{aligned} \tag{33}$$

The latter is exactly the eigen-problem of Laplacian Eigenmap [4]. Recall the eigen-problem of Locality Preserving Projection (LPP) [17] which is as follows:

$$XLX^T\mathbf{a} = \lambda XDX^T\mathbf{a} \tag{34}$$

From Theorem (1) and (2), we see that the fourth step of our algorithm provides a regularized locality preserving projection solution.

Clustering by using the top eigenvectors of a matrix derive from the distance between points (like W in our algorithm), so called *spectral clustering methods*, recently received a lot of interests [22, 32]. But

despite their empirical successes, there is still no theoretical proof on which eigenvectors to use and how to derive clusters from them. Those eigenvectors also play a key role in our algorithm, we provide here a informal analysis on how many eigenvectors we should use when there exist c classes. Our analysis is based on graph partitioning.

We consider the problem of dividing the graph $\mathcal{G}$ into c disjoint subgraphs, $\mathcal{G}_1, \cdots, \mathcal{G}_c$, such that $\mathcal{G} = \mathcal{G}_1 \cup \cdots \cup \mathcal{G}_c$ and $\mathcal{G}_k \cap \mathcal{G}_p = \emptyset,\ k \neq p$. The min-cut criteria can be stated as follows:

$$\min_{\mathcal{G}_1,\cdots,\mathcal{G}_c} \left(\sum_{i\in\mathcal{G}_1} \sum_{j\in\mathcal{G}_2} W_{ij} + \cdots + \sum_{i\in\mathcal{G}_{c-1}} \sum_{j\in\mathcal{G}_c} W_{ij} \right) \tag{35}$$

where W is the weight matrix of graph $\mathcal{G}$. Now, let's recall the objective function of Lapalcian Eigenmaps [4], which is also the objective function for many spectral clustering algorithms::

$$\min \sum_{ij} \|Y_i - Y_j\|^2 W_{ij} \tag{36}$$

It is easy to verify that Eqn (36) gives the same solution of Eqn (35) as long as Y_i's satisfy two conditions: (1) $\|Y_i - Y_j\| = 0$ if $\mathbf{x}_i$ and $\mathbf{x}_j$ belong to the same subgraph; (2) $\|Y_i - Y_j\| = d$, d is a constant for any $\mathbf{x}_i$ and $\mathbf{x}_j$ as long as they belong to different subgraph. To fulfil such requirement, we can immediately see the label vector Y_i's reside on the vertices of the standard $(c-1)$-simplex, *i.e.*, Y_i should be at least $c-1$ dimension. Let $Y = [Y_1, \cdots, Y_m]^T$. By relaxing the elements of Y to take real values. The column vectors of optimal Y will be given by the $c-1$ smallest eigenvectors of eigen-problem $L\mathbf{y} = \lambda D\mathbf{y}$ [4].

The above informal analysis shows that we need at least $c-1$ (after removing the first all ones trivial eigenvector) smallest eigenvectors of L to reveal the c class structure. Such analysis is consistent with previous study on spectral clustering [22]. Also, our analysis in the previous section shows that when the label information are available, we can use exactly $c-1$ eigenvectors to reveal the class structure.

It is also important to note that many spectral clustering algorithms [1, 22, 32] and spectral dimensionality reduction algorithm [4] do not have explicit embedding function, thus can not be applied to unseen data. Similar to LPP, SR provides natural out-of-sample extensions to Laplacian Eigenmap [4] and several other spectral clustering algorithms [22, 32]. Moreover, the computation of our approach can be more efficient than ordinary LPP [17].

4.3 Spectral Regression in Semi-Supervised Setting

For SR in semi-supervised setting, we have the following theorem.

Theorem 5 *Let $\boldsymbol{y}$ be one of the first c largest eigenvectors of eigen-problem (17), $\boldsymbol{y} = [y_1, y_2, \cdots, y_m]^T$. Suppose two labeled points $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ share the same label. For any $\eta > 0$, there exists δ (parameter in the second step of our algorithm) such that $|y_i - y_j| \leq \eta$.*

Proof Without loss of generality, we assume that the labeled data points $\{\mathbf{x}_1, \cdots, \mathbf{x}_l\}$ are ordered according to their labels, $\mathbf{x}_i$ and $\mathbf{x}_j$ belong to the first class and there is l_1 labeled points in the first class. The eigenvectors are normalized , *i.e.*, $\|\mathbf{y}\| = 1$.

The first l_1 elements in the i-th and j-th rows of W are $1/l_1$ and the next $m - l_1$ elements are either zero or $\delta \cdot s(i,j)$ as defined in Eq. (16). It is easy to check the rank of W is grater than c. Thus, $\mathbf{y}$ is an eigenvector with non-zero eigenvalue, $W\mathbf{y} = \lambda D\mathbf{y}$. We have

$$y_i = \frac{1}{\lambda}\frac{1}{D_{ii}}\sum_{q=1}^{m} W_{iq}y_q = \frac{1}{\lambda}\frac{1}{D_{ii}}\left(\frac{1}{l_1}\sum_{q=1}^{l_1} y_q + \sum_{q=l_1+1}^{m} W_{iq}y_q\right)$$
$$(1 \le i \le l_1)$$

Since $s(i,j) \le 1,\ (\forall\ i,j)$ and $\|\mathbf{y}\| = 1$, we have

$$\begin{aligned}
&W_{iq} \le \delta, \quad (q = l_1 + 1, \cdots, m),\\
&W_{jq} \le \delta, \quad (q = l_1 + 1, \cdots, m),\\
&D_{ii} = \sum_q W_{iq} \le 1 + (m - l_1)\delta,\\
&D_{jj} = \sum_q W_{jq} \le 1 + (m - l_1)\delta,\\
&|y_i| \le 1, \quad (i = 1, \cdots, m).
\end{aligned}$$

Thus,

$$\begin{aligned}
|y_i - y_j| &= \left|\frac{1}{\lambda}\frac{1}{l_1}\sum_{q=1}^{l_1} y_q\left(\frac{1}{D_{ii}} - \frac{1}{D_{jj}}\right)\right.\\
&\quad \left. + \frac{1}{\lambda}\left(\frac{1}{D_{ii}}\sum_{q=l_1+1}^{m} W_{iq}y_q - \frac{1}{D_{jj}}\sum_{q=l_1+1}^{m} W_{jq}y_q\right)\right|\\
&\le \frac{1}{\lambda}\frac{1}{l_1}\sum_{q=1}^{l_1}|y_q|\left|\frac{D_{jj} - D_{ii}}{D_{ii}D_{jj}}\right|\\
&\quad + \frac{1}{\lambda}\frac{1}{D_{ii}}\sum_{q=l_1+1}^{m} W_{iq}|y_q| + \frac{1}{\lambda}\frac{1}{D_{jj}}\sum_{q=l_1+1}^{m} W_{jq}|y_q|\\
&\le \frac{1}{\lambda}\frac{1}{l_1}\sum_{q=1}^{l_1}|y_q|\Big(2(m - l_1)\delta\Big) + \frac{2}{\lambda}\sum_{q=l_1+1}^{m}|y_q|\delta\\
&= \left(\frac{2(m - l_1)}{\lambda l_1}\sum_{q=1}^{l_1}|y_q| + \frac{2}{\lambda}\sum_{q=l_1+1}^{m}|y_q|\right)\delta\\
&\le \left(\frac{4(m - l_1)}{\lambda}\right)\delta
\end{aligned}$$

Given $\eta > 0$, we choose $\delta = \eta\lambda/4(m - l_1)$. Thus $|y_i - y_j| \le \eta$.

The above theorem shows that when δ is sufficiently small, the responses of the labeled same class points can be as close as possible. This property is reasonable and important since for classification, either supervised or semi-supervised, we always expect that the responses of the samples in the same class are close to each other.

5 Computational Complexity Analysis

In this section, we provide a computational complexity analysis of SR. Our analysis considers both time complexity and memory cost. The term *flam*, a compound operation consisting of one addition and one multiplication, is used for presenting operation counts [28].

To get a better understanding of the benefits SR achieves, we first provide a computational complexity analysis of LDA and LPP.

5.1 Complexity Analysis of LDA and LPP

Both LDA and LPP need to solve the eigen-problem in Eqn. (12) with different W. LDA can get some computational benefits from the special structure of W as shown in the following equations (We have $D_{LDA} = I$).

$$\begin{aligned} & XW_{LDA}X^T\mathbf{a} = \lambda XX^T\mathbf{a} \\ \Rightarrow \quad & U\Sigma V^T W_{LDA} V\Sigma U^T\mathbf{a} = \lambda U\Sigma\Sigma U^T\mathbf{a} \\ \Rightarrow \quad & \Sigma^{-1}U^TU\Sigma V^T W_{LDA} V\Big(\Sigma U^T\mathbf{a}\Big) = \lambda\Sigma^{-1}U^TU\Sigma\Big(\Sigma U^T\mathbf{a}\Big) \\ \Rightarrow \quad & V^T W_{LDA} V\mathbf{b} = \lambda\mathbf{b} \end{aligned} \tag{37}$$

$V \in \mathbb{R}^{m\times d}$ is right singular matrix of X and d is the rank of X. The i-th row vector of V corresponds to the data point $\mathbf{x}_i$ and we denote it as $\mathbf{z}_i$, $V = [\mathbf{z}_1, \cdots, \mathbf{z}_m]^T$. Let $\mathbf{z}_i^{(k)}$ denote the row vector of V which corresponds to $\mathbf{x}_i^{(k)}$. Define $\boldsymbol{\nu}^{(k)} = \frac{1}{l_k}\sum_{i=1}^{l_k}\mathbf{z}_i^{(k)}$ and $H = [\sqrt{l_1}\boldsymbol{\nu}^{(1)}, \cdots, \sqrt{l_c}\boldsymbol{\nu}^{(c)}] \in \mathbb{R}^{d\times c}$. Inspired by Eqn. (29), we have

$$\begin{aligned} V^T W_{LDA} V =& \sum_{k=1}^{c}\frac{1}{l_k}\left(\sum_{i=1}^{l_k}\mathbf{z}_i^{(k)}\sum_{i=1}^{l_k}(\mathbf{z}_i^{(k)})^T\right) \\ =& \sum_{k=1}^{c} l_k\ \boldsymbol{\nu}^{(k)}(\boldsymbol{\nu}^{(k)})^T \\ =& HH^T \end{aligned} \tag{38}$$

The above algebraic steps show that the LDA projective functions can be obtained by the SVD decomposition of X and calculating the eigenvectors of HH^T.

It is easy to check that the left singular vectors of X (column vectors of U) are the eigenvectors of XX^T and the right singular vectors of X (column vectors of V) are the eigenvectors of X^TX [29]. Moreover, if U or V is given, then we can recover the other via the formula $XV = U\Sigma$ and $U^TX = \Sigma V^T$. In fact, the most efficient SVD decomposition algorithm (i.e. *cross-product*) applies this strategy [29]. Specifically, if $m \geq n$, we compute the eigenvectors of XX^T, which gives us U and can be used to recover V; If $m < n$, we compute the eigenvectors of X^TX, which gives us V and can be used to recover U. Since the matrix H is of size $r \times c$, where r is the rank of X and c is the number of classes. In most of the cases, r is close to $\min(m, n)$ which is far larger than c. Thus, comparing to directly calculate

the eigenvectors of HH^T, compute the eigenvectors of H^TH then recover the eigenvectors of HH^T can achieve a significant saving.

When $m \geq n$, the calculation of XX^T requires $\frac{1}{2}mn^2$ flam; Computing the eigenvectors of XX^T requires $\frac{9}{2}n^3$ flam [29, 11]; Recovering V from U requires mn^2 flam by assuming X is of full rank; Computing the eigenvectors of HH^T requires $\frac{1}{2}nc^2 + \frac{9}{2}c^3 + nc^2$ flam; Finally, calculating $\mathbf{a}$'s from $\mathbf{b}$'s requiring n^2c. When $m < n$, we have the similar analysis. We conclude that the time complexity of LDA measured by flam is

$$\frac{3}{2}mnt + \frac{9}{2}t^3 + \frac{3}{2}tc^2 + \frac{9}{2}c^3 + t^2c$$

where $t = \min(m, n)$. Considering $c \ll t$, the time complexity of LDA can be written as $\frac{3}{2}mnt + \frac{9}{2}t^3 + O(t^2)$.

For LPP, we have (For simplicity, we use W, instead of W_{LPP}.)

$$\begin{aligned} & XWX^T\mathbf{a} = \lambda XDX^T\mathbf{a} \\ \Rightarrow \quad & \Big(XD^{1/2}\Big)\Big(D^{-1/2}WD^{-1/2}\Big)\Big(XD^{1/2}\Big)^T\mathbf{a} = \lambda\Big(XD^{1/2}\Big)\Big(XD^{1/2}\Big)^T\mathbf{a} \\ \Rightarrow \quad & \bar{U}\bar{\Sigma}\bar{V}^T\bar{W}\bar{V}\bar{\Sigma}\bar{U}^T\mathbf{a} = \lambda\bar{U}\bar{\Sigma}\bar{\Sigma}\bar{U}^T\mathbf{a} \\ \Rightarrow \quad & \bar{V}^T\bar{W}\bar{V}\mathbf{b} = \lambda\mathbf{b} \end{aligned} \tag{39}$$

where $XD^{1/2} = \bar{U}\bar{\Sigma}\bar{V}^T$ is the SVD decomposition and $\bar{W} = D^{-1/2}WD^{-1/2}$. The $\bar{W}$ does not have the special structure as W_{LDA} and the approach described in Eqn. (38) can not be applied. We need to directly calculate the eigenvectors of $\bar{V}^T\bar{W}\bar{V}$. With $\frac{9}{2}t^3$ flam, all the t eigenvectors can be calculated. If only d projective functions are needed, the Lanczos algorithm can be used to iteratively compute the first d eigenvectors within k_1dt^2 flam, where k_1 is the number of iterations[4] in Lanczos [29].

The affinity graph construction step (calculating the W) in LPP is a necessary step in all the unsupervised spectral embedding algorithms. If we use p-nearest neighbor graph, the time complexity is around $\frac{1}{2}m^2n + 2mn + m^2\log m$ flam. $\frac{1}{2}m^2n + 2mn$ is used to calculate the pairwise distances and $m^2\log m$ is used for m times sorting[5]. Overall, the time complexity of LPP measured by flam is

$$m^2(\frac{1}{2}n + \log m) + \frac{3}{2}mnt + \frac{9}{2}t^3 + \min(\frac{9}{2}t, k_1d)t^2 + O(t^2)$$

Besides the data matrix X, both LDA and LPP need to store the left and right singular matrices of X (or $XD^{1/2}$), which are both dense matrices. Thus the memory cost of LDA (LPP) is at least

$$ms + mt + nt$$

where s is the average number of nonzero features for one sample ($s \leq n$).

[4]Lanczos algorithm converges very fast, 20 iterations are usually enough to achieve a satisfactory precision [29].

[5]There exist more efficient algorithms to obtain the p-nearest neighbors in stead of sorting the m numbers. We will not discuss this since it is beyond the scope of this paper.

5.2 Complexity Analysis of SR

SR uses regularized least squares to find the projective functions, which is a necessary step in both supervised and unsupervised cases. Thus, we begin our analysis with analyzing this step.

When n is not very large, the regularized least squares problem in Eqn. (18) can be solved by directly solving the linear equations system in Eqn. (19). The calculation of XX^T requires $\frac{1}{2}mn^2$ flam. Since the matrix $XX^T + \alpha I$ is positive definite, using Gaussian Elimination to solve the linear equations system in Eqn. (19) costs $\frac{1}{6}n^3$ flam [28].

For large scale high dimensional data, the regularized least squares problem in Eqn. (18) can be efficiently solved by iterative algorithm LSQR which is designed to solve large scale sparse linear equations and least squares problems [24]. In each iteration, LSQR needs to compute two matrix-vector products in the form of $X\mathbf{p}$ and $X^T\mathbf{q}$. The remaining work load of LSQR in each iteration is $3m+5n$ flam [23]. Thus, the time cost of LSQR in each iteration is $2mn + 3m + 5n$. If LSQR stops after k_2 iterations[6], the time cost is $k_2(2mn + 3m + 5n)$. Finally, the total time cost for d projective functions is $dk_2(2mn + 3m + 5n)$. Besides data matrix X, LSQR needs $m + 2n$ additional memory [23]. Finally, the memory cost in this step is $mn + m + 2n + dn$, with dn to store the projective functions.

In supervised case, the eigen-problem in third step of SR is trivial and we can directly obtain those $c - 1$ eigenvectors. The cost of this step is mainly the cost of Gram-Schmidt method, which requires $(mc^2 - \frac{1}{3}c^3)$ flam and $mc + c^2$ memory [28].

In unsupervised case, the affinity graph construction step is same as we analyzed before. Since the p-nearest neighbor graph matrix W is sparse (has around mp non-zero entries), we can use Lanczos algorithm to compute the first d eigenvectors within $dk_1m(p + 8)$ flam [29]. The memory requirement of this step is simply the memory to store W and d eigenvectors.

5.3 Summary

We summarize our complexity analysis results in Table 2. We assume $m \gg c$ and only show the dominant part of the time and memory costs for simplicity. The main conclusions include:

- In supervised case:
  - ◇ LDA has cubic-time complexity with respect to $\min(m, n)$. Moreover, the left and right singular vector matrices of X, which are required to be stored in memory, are both dense. When both m and n are large, it is not feasible to apply LDA.
  - ◇ SR has linear-time complexity with respect to both m and n. It only has very small additional memory requirement besides data matrix X. Thus, SR can be easily scaled to high dimensional large data sets.
  - ◇ The computational complexity analysis clearly shows the advantages of using SR instead of directly applying LDA.

[6]LSRQ converges very fast [24]. In our experiments, 20 iterations are enough.

Table 2: Computational complexity of LDA, LPP and SR

Time complexity (operation counts, *flam*)				
Algorithm		Graph Construction	Responses Generation	Embedding Functions
Supervised	LDA	–	–	$\frac{3}{2}mnt + \frac{9}{2}t^3$
	SR		mc^2	$2ck_2ms + 5ck_2n$
Unsupervised	LPP	$m^2(s + \log m)$	–	$\frac{3}{2}mnt + \frac{9}{2}rt^3 + \min(\frac{9}{2}t, dk_1)t^2$
	SR		$dk_1m(p + 8)$	$2dk_2ms + 5dk_2n$
Memory cost				
Algorithm				
Supervised	LDA	$ms + (m + n)t + nc$		
	SR	$ms + mc + nc$		
Unsupervised	LPP	$ms + mp + (m + n)t + nd$		
	SR	$ms + mp + md + nd$		

m: the number of data samples
n: the number of features
t: $\min(m, n)$
s: the average number of nonzero features for one sample ($s \leq n$)
c: the number of classes (LDA and SR will produce $c - 1$ projective functions)
d: the number of dimensions (projective functions) required in LPP and SR
p: the number of nearest neighbors
k_1: the number of iterations in Lanczos
k_2: the number of iterations in LSQR

- In unsupervised case:
  - ◇ The graph construction step is unavoidable for all the spectral graph embedding approaches. If the same graph is used, the computational cost on this step can be neglected when we compare the different algorithms.
  - ◇ The popular manifold learning algorithm (*e.g.*, LLE, Isomap, Laplacian Eigenmaps) only compute the embedding results of the training data, which is exactly the responses generation step of SR. SR uses regression to find the projective functions with the additional linear-time complexity cost (with respect to both m and n) and almost no additional memory requirement.
  - ◇ Those linear (kernel) extension approaches (*e.g.*, LPP, NPE, Kernel Eigenmaps) directly calculate the projective functions by solving dense eigen-problems. They require additional cubic-time complexity cost (with respect to $\min(m, n)$) and $(m+n)\cdot\min(m, n)$ memory cost. When both m and n are large, it is infeasible to apply these approaches.
- In both cases:
  - ◇ In many real problems, the data matrix is sparse. However, LDA and LPP need the **complete** SVD decomposition, which can not get any benefit from the sparseness of the data matrix.

Moreover, the left and right singular matrices are both dense. They can not be fit into the memory when both m and n are large.

- ◇ As shown in Table (2), SR can fully explore the sparseness of the data matrix and gain significant computational saving on both time and memory. SR can successfully applied as long as the data matrix X can be fit into the memory.
- ◇ Even the data matrix X is too large to be fit into the memory, SR can still be applied with some reasonable disk I/O. This is because in each iteration of LSQR, we only need to calculate two matrix-vector products in the form of $X\mathbf{p}$ and $X^T\mathbf{q}$, which can be easily implemented with X and X^T stored on the disk.

6 Experiments

In this section, several experiments on classification and semi-supervised classification were performed to show the effectiveness and efficiency of our proposed algorithm. All of our experiments have been performed on a P4 3.20GHz Windows XP machines with 1GB memory.

6.1 Experiments on Supervised Learning

6.1.1 Face Recognition on PIE

Dimensionality reduction has been widely used in appearance-based face recognition. Two of the most well known approaches include *Eigenface* [31] and *Fisherface* [3]. Eigenface is based on Principal Component Analysis (PCA) and Fisherface is based on Linear Discriminant Analysis (LDA). Fisherface performs PCA first to guarantee the nonsingularity of with-class scatter matrix S_w which is essentially same as the SVD approach we discussed in Section 2. The ordinary Regularized Linear Discriminant Analysis (RLDA) [9] solves the singularity problem by adding some constant values to the diagonal elements of S_w, as $S_w + \alpha I$, for some $\alpha > 0$. In this experiment, we compared SR with these three approaches.

All the algorithms are tested on the CMU PIE face database[7], which contains 68 subjects with 41,368 face images as a whole. The face images were captured under varying pose, illumination and expression. We choose the five near frontal poses (C05, C07, C09, C27, C29) and use all the images under different illuminations and expressions, thus we get 170 images for each individual. All the face images are manually aligned and cropped. The cropped images are 32×32 pixels, with 256 gray levels per pixel. The features (pixel values) are then scaled to [0,1] (divided by 256). For each individual, $l(= 10, 20, 30, 40, 50, 60)$ images are randomly selected for training and the rest are used for testing.

The nearest neighbor classifier is applied in the original face image space (Baseline) and PCA, LDA, RLDA, SR subspace. For LDA, RLDA and SR, the dimension of the subspace is $c - 1$ (=67), where c is the number of categories. For PCA, we tested its performance with all the possible dimensions and report the best result. Notice that there is a parameter α which controls smoothness of the estimator

[7] http://www.ri.cmu.edu/projects/project_418.html

Table 3: Recognition error rates on PIE (mean±std-dev%)

Train Size	Baseline	Eigenface (PCA)	Fisherface (LDA)	RLDA	SR
10×68	64.8±0.7	64.8±0.7	29.7±1.3	19.1±1.2	19.5±1.3
20×68	48.6±0.7	48.6±0.7	20.5±0.8	10.9±0.7	10.8±0.7
30×68	37.9±0.6	37.9±0.6	10.9±0.5	8.7±0.7	8.4±0.7
40×68	29.9±0.6	29.9±0.6	8.2±0.4	7.2±0.5	6.9±0.4
50×68	23.9±0.6	23.9±0.6	7.2±0.4	6.6±0.4	6.3±0.4
60×68	19.6±0.6	19.6±0.6	6.4±0.3	6.0±0.3	5.7±0.2

Table 4: Computational time on PIE (s)

Train Size	Eigenface (PCA)	Fisherface (LDA)	RLDA	SR
10×68	2.527	4.291	7.725	0.468
20×68	7.012	7.626	7.828	0.685
30×68	7.329	7.887	7.908	0.903
40×68	7.558	8.130	8.278	1.126
50×68	7.693	8.377	8.414	1.336
60×68	7.949	8.639	8.654	1.573

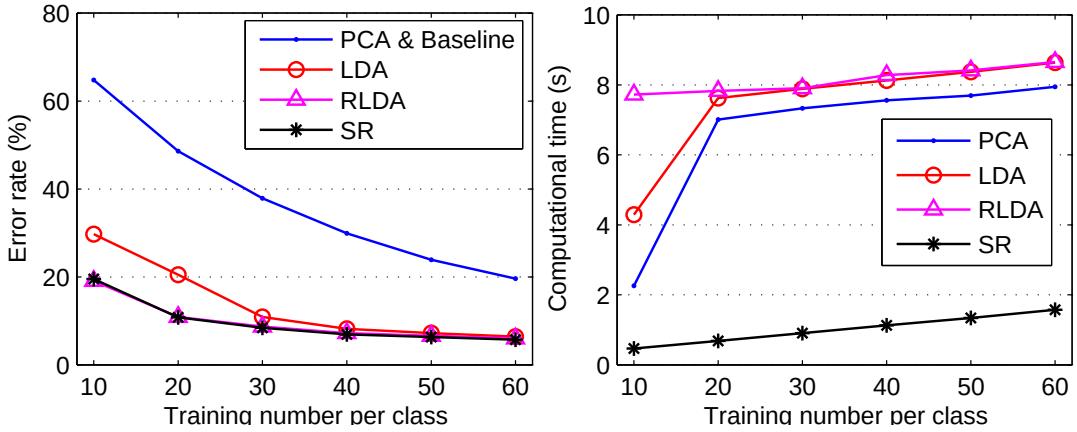


Figure 1: Error rate and computational time as functions of number of training samples per class on PIE.

in both RLDA and SR. We simply set the value of α as 1, and the effect of parameter selection will be discussed later.

The recognition error rate as well as the the running time (second) of computing the projection functions for each method are reported on the Table (3) and (4). These results are also showed in the Figure (1). For each given l (the number of training samples per class), we average the results over 20 random splits and report the mean as well as the standard deviation.

The main observations from the performance comparisons include:

- The eigenface approach (PCA) fails to gain any improvement over the baseline method. This is probably due to the fact that PCA is unsupervised and does not encode discriminating information.

- The fisherface approach (LDA) seeks the projective functions which are optimal on the training set. It does not consider the possible overfitting in small sample size case. Both RLDA and SR are regularized versions of LDA. The Tikhonov regularizer is used to control the model complexity. When the number of training samples are small, RLDA and SR achieve significantly better performance than LDA, which suggests that overfitting is a very crucial problem which should be addressed in LDA model.

- Both LDA and RLDA need SVD decomposition of the data matrix which is computationally expensive. While SR only needs to solve $c-1$ regularized least squares problems which are very efficient. This nice property makes it possible to apply SR to high dimensional large data sets.

6.1.2 Text Categorization on 20Newsgroups

The popular 20 Newsgroups[8] is a data set collected and originally used for document classification by Lang [19]. The "bydate" version is used in our experiment. The duplicates and newsgroup-identifying headers are removed which leaves us 18,941 documents, evenly distributed across 20 classes. This corpus contains 26214 distinct terms after stemming and stop word removal. Each document is then represented as a term-frequency vector and normalized to 1.

The most well known dimensionality reduction algorithm for text processing is Latent Semantic Indexing (LSI) [8] which is essentially similar to PCA. The LDA obviously can also be used for text classification (with the necessary SVD pre-processing). However, the ordinary RLDA can not be applied because it needs to solve the generalized eigen-problem with matrices size $n \times n$, where n is the number of features. In our case, $n = 26214$ and such matrices can not be fit into memory.

The experimental setting is the same as before, so as all the parameters. The only difference is that it is not possible to enumerate all the dimensions in LSI, and we tested its performance with dimension 50, 100 and 500.

The data set was randomly split into training and testing sets. In order to examine the effectiveness of different algorithms with different size of the training set, we ran several tests that the training set contains 5%, 10%, 20%, 30%, 40% and 50% documents. For each test, we averaged the results over 20 random splits. The categorization error rates as well as the dimensionality reduction time of all the methods are shown in Table (5), (6) and Figure (2).

Similar to PCA, LSI is also unsupervised and does not have discriminating power. It is not a good choice in dimensionality reduction for text categorization. The error rate in LSI subspace is even larger than that in original document space. Both LDA and SR are supervised. They search for the subspace in which the different class samples are far from each other while same class samples are close to each other. As a result, classification in the LDA (SR) subspace achieves significant better performance than in original document space. Again, as a regularized version of LDA, SR achieves better performance than LDA, which verifies the effectiveness of regularization for classification [15].

LDA has the cubic-time complexity with respect to $\min(m, n)$ and it also needs the memory to store

[8] http://people.csail.mit.edu/jrennie/20Newsgroups/

Table 5: Categorization error rates on 20Newsgroups (mean±std-dev%)

Train Size	Baseline	LSI 50	LSI 100	LSI 500	LDA*	RLDA*	SR
5%	41.8±1.3	54.5±0.8	50.1±1.2	43.7±0.8	28.0±0.6	–	27.3±0.5
10%	35.7±0.5	50.6±1.1	46.2±0.7	39.1±0.6	22.7±0.6	–	21.3±0.5
20%	30.3±0.2	46.5±0.7	42.4±0.4	35.5±0.8	–	–	16.0±0.3
30%	27.2±0.3	44.0±0.5	40.1±0.4	33.8±0.4	–	–	13.8±0.2
40%	24.6±0.3	42.5±0.3	38.1±0.3	31.3±0.4	–	–	12.4±0.2
50%	22.4±0.4	41.4±0.4	36.7±0.3	29.4±0.3	–	–	11.4±0.2

Table 6: Computational time of LSI, LDA and SR (s)

Train Size	LSI 50	LSI 100	LSI 500	LDA*	RLDA*	SR
5%	6.53	11.58	19.52	24.54	–	16.47
10%	9.91	28.12	61.29	81.08	–	19.23
20%	16.45	44.13	138.9	–	–	22.93
30%	22.13	58.13	301.8	–	–	26.84
40%	27.80	70.86	518.7	–	–	31.24
50%	32.87	82.60	708.4	–	–	36.51

*LDA (RLDA) can not be applied as the size of training set increases due to the memory

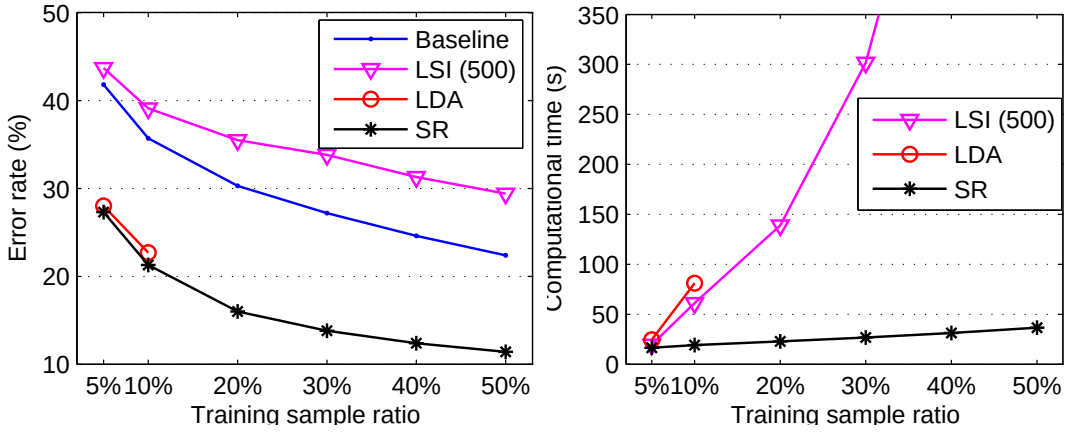
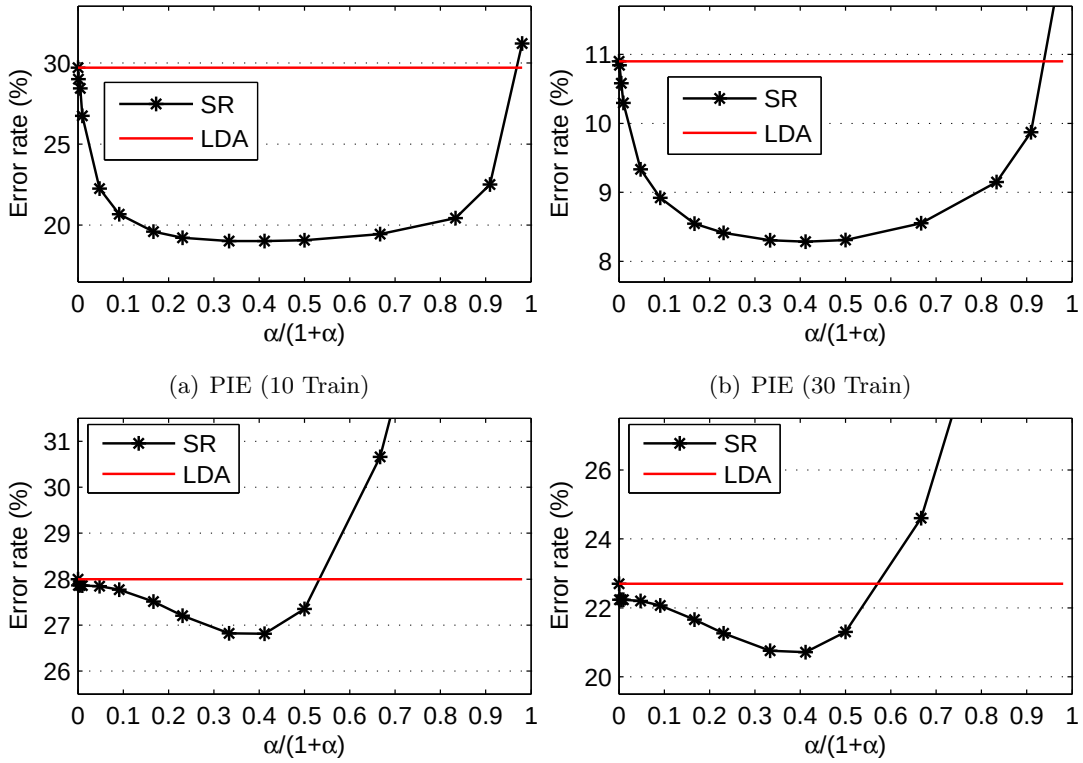


Figure 2: Error rate and computational time as functions of number of training samples per class on 20Newsgroups.

the left and right singular vector matrices which are both dense. As the size of the training set increases, LDA can not be applied due to the memory limit. SR has linear-time complexity with respect to both m and n. Moreover, it can fully explore the sparseness of the data matrix and gain significant computational saving on both time and memory. These properties make SR an efficient and effective dimensionality reduction algorithm for text processing tasks (high dimensional large scale data).



(a) PIE (10 Train)

(b) PIE (30 Train)

(c) 20Newsgroup (5% Train)

(d) 20Newsgroup (10% Train)

Figure 3: Model selection of SR with α on PIE (a, b) and 20Newsgroups (c, d). The curve shows the test error of SR with respect to $\alpha/(1+\alpha)$. The other line shows the test error of LDA. It is clear that SR can achieve significantly better performance than LDA over a large range of α.

6.1.3 Model Selection with α

The only parameter in SR (supervised mode) is the regularization parameter $\alpha \geq 0$ which controls the smoothness of the estimator. Our theoretical analysis shows that SR gives the LDA solution as α decreases to zero. In the previous experiments, we empirically set it to be 1. In this subsection, we try to examine the impact of parameter α on the performance of SR.

Figure (3) shows the performance of SR as a function of the parameter α. For convenience, the X-axis is plotted as $\alpha/(1+\alpha)$ which is strictly in the interval $[0, 1]$. It is easy to see that SR can achieve significantly better performance than LDA over a large range of α. Thus, the parameter selection is not a very crucial problem in SR algorithm. In reality, we can use cross validation to select this parameter or simply choose a value between 0.1 and 1.

6.2 Experiments on Semi-Supervised Learning

In the following subsections, we try to examine the performance of our algorithm in semi-supervised situation. In many of previous experiments for semi-supervised learning, the experimental setting is

transductive (*e.g.*, [34, 35]), that is, both the training and test set (without label information) are available during the learning process. In reality, a more natural setting for semi-supervised learning is as follows. The available training set contains both labeled and unlabeled examples, and the testing set is not available during the training phrase, which we refer here as *semi-supervised setting*. In this case, many of recently proposed graph based semi-supervised learning algorithms can not be applied since they do not have out-of-sample extension. SR learns the embedding functions which are defined everywhere, thus, our algorithm can be applied in both of these two settings.

6.2.1 Handwritten Digit Recognition

The MNIST handwritten digit database[9] was used in this experiment. The public MNIST database has a training set of 60,000 samples (denoted as set A), and a testing set of 10,000 samples (denoted as set B). In our experiment, we take the first 2,000 samples from the set A as our training set and the first 2,000 samples from the set B as our test set. In both training and test sets, each digit has around 200 samples.

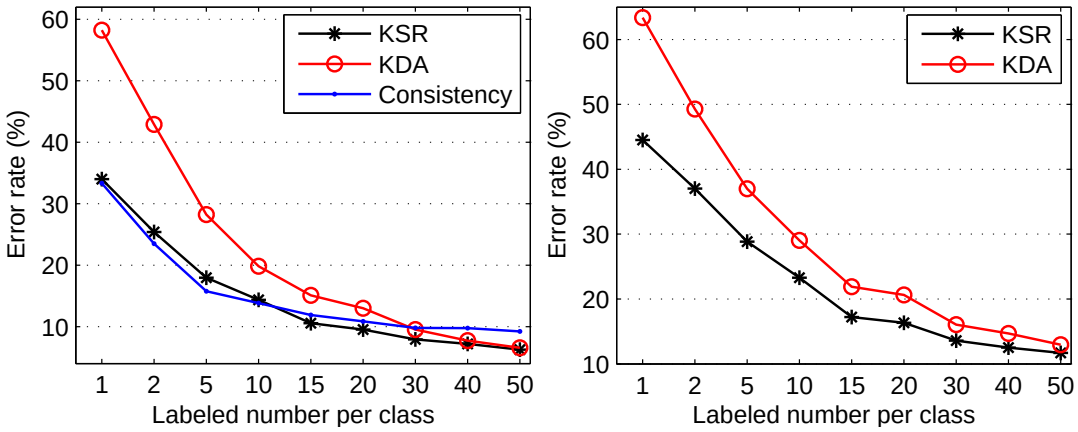
A random subset with l $(= 1, 2, 5, \cdots, 40, 50)$ samples per digit from the training set were labeled and the rest were left unlabeled. Predicting the label of those unlabeled samples is a transductive setting and predicting the label of samples in test set is a semi-supervised setting. In the transductive setting, three algorithms are compared. They are: (1) Kernel Discriminant Analysis (KDA) [2], which uses labeled data only; (2) KSR, which uses both labeled and unlabeled data; (3) Consistency [34], which uses both labeled and unlabeled data. In the semi-supervised setting, the first two algorithms are compared. For KDA and KSR, the classification is based on the distances of a test example to the class centroids in the KDA (KSR) subspace. We average the results over 20 random selection for each given l.

The p is set to be 5 for the p-nearest neighbor graph over all the training samples in KSR. This graph is also used in the Consistency algorithm. The values of both parameters α and δ in KSR were set to 0.1. The same Gaussian RBF kernel with width 1 is used in both KDA and KSR. The parameter in Consistency algorithm [34] was tuned to achieve the optimal performance.

Figure 4(a) and 4(b) show the performance of different algorithms in the transductive setting and semi-supervised setting respectively. We have the following observations: (a) All these algorithms can take advantage of more labeled samples, which is important to the real-world classification task. (b) KSR can make efficient use of both labeled and unlabeled samples to discover the intrinsic discriminant structure in the data, thus achieves consistent improvement over KDA which is purely supervised. (c) KSR achieves comparable performance with Consistency algorithm in transductive setting. The latter is one of the state-of-the-art semi-supervised learning algorithms. Moreover, KSR can also be applied in semi-supervised setting which gives us more flexibility.

Table 7: Performance comparisons on MNIST (mean±std-dev%)

Labeled Size	Unlabeled Error			Test Error	
	KDA	KSR	Consistency	KDA	KSR
1×26	58.3±4.9	34.0±4.9	33.3±5.0	63.4±5.3	45.5±4.3
2×26	42.9±3.1	25.4±3.3	23.5±4.1	49.3±3.6	37.0±5.6
5×26	28.3±2.6	17.9±2.8	15.8±2.1	34.0±2.7	29.8±4.4
10×26	19.8±1.1	14.4±2.4	14.2±1.1	29.1±2.1	24.3±3.9
15×26	15.1±1.4	10.6±0.8	11.9±0.7	21.9±1.9	17.2±1.6
20×26	13.0±0.8	9.5±0.9	10.9±0.6	20.6±2.1	16.3±1.6
30×26	9.5±0.7	7.9±0.4	9.8±0.5	16.1±1.0	13.6±0.8
40×26	7.7±0.4	7.2±0.5	9.7±0.6	14.7±0.8	12.5±0.7
50×26	6.5±0.3	6.2±0.4	8.9±0.7	13.0±0.9	11.7±0.5



(a) Unlabeled error

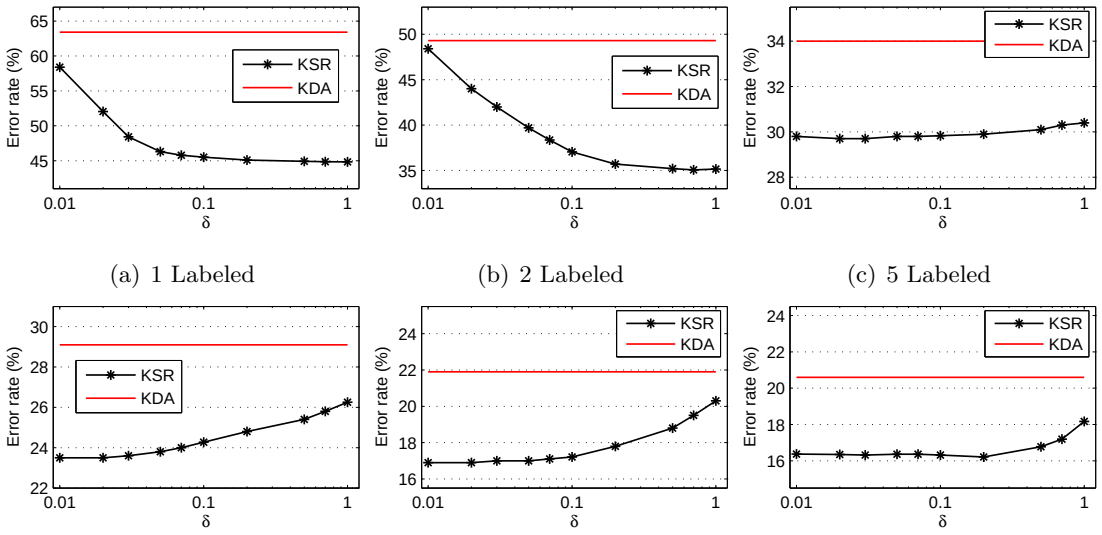
(b) Test error

Figure 4: Error rate as a function of number of labeled samples.

6.2.2 Model Selection with δ

In semi-supervised setting, the essential parameter in SR is the δ ($0 < \delta \leq 1$) which adjusts the weight between supervised information and unsupervised neighbor information. Figure (5) shows the performance of KSR as a function of the parameter δ. When the number of labeled samples is extremely small (1 or 2 labeled samples per class), the optimal δ tends to be large (larger than 0.1); With the number of labeled samples increases, the optimal δ decreases. In all cases, KSR can achieve significantly better performance than KDA over a large range of δ.

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



(a) 1 Labeled (b) 2 Labeled (c) 5 Labeled

(d) 10 Labeled (e) 15 Labeled (f) 20 Labeled

Figure 5: Model selection of SR with δ on MNIST. The curve shows the test error of KSR with respect to δ. The other line shows the test error of KDA. When the number of labeled samples is extremely small (1 or 2 labeled samples per class), the optimal δ tends to be large (larger than 0.1); With the number of labeled samples increases, the optimal δ decreases. In all cases, KSR can achieve significantly better performance than KDA over a large range of δ.

7 Conclusion

In this paper, we propose a new dimensionality reduction algorithm called *Spectral Regression* (SR). It is based on the same variational principle that gives rise to the Laplacian Eigenmap [4]. As a natural extension of several recent nonparametric techniques for global nonlinear dimensionality reduction such as [26, 30, 4], SR aims at learning an embedding function (either linear or in RKHS) which is defined everywhere (and therefore on novel test data points). It casts the problem of learning an embedding function into a regression framework which facilitates both efficient computation and the use of regularization techniques. The computational complexity analysis illustrates the advantage of SR over other linear or kernel extensions of LLE and Laplacian Eigenmap [17, 6, 16].

By using the affinity graph to model both label and local neighborhood information, SR can make efficient use of both labeled and unlabeled points to discover the intrinsic discriminant structure in the data. Our theoretical analysis linked our algorithm to LDA [14] and LPP [17] in supervised and unsupervised cases. The experimental results on classification and semi-supervised classification demonstrate the effectiveness and efficiency of our algorithm.

Our approach provides a general framework for learning a function (either linear or in RKHS) in graph embedding approaches. With the specific affinity graph, SR can provide a natural out-of-sample extension of many spectral embedding algorithms like LLE, Isomap, Laplacian Eigenmaps and spectral clustering algorithms [22].

References

[1] C. J. Alpert, A. B. Kahng, and S.-Z. Yao. Spectral partitioning with multiple eigenvectors. *Discrete Applied Mathematics*, 90:3–26, 1999.

[2] G. Baudat and F. Anouar. Generalized discriminant analysis using a kernel approach. *Neural Computation*, (12):2385–2404, 2000.

[3] P. N. Belhumeur, J. P. Hepanha, and D. J. Kriegman. Eigenfaces vs. fisherfaces: recognition using class specific linear projection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(7):711–720, 1997.

[4] M. Belkin and P. Niyogi. Laplacian eigenmaps and spectral techniques for embedding and clustering. In *Advances in Neural Information Processing Systems 14*, pages 585–591. MIT Press, Cambridge, MA, 2001.

[5] Y. Bengio, J.-F. Paiement, P. Vincent, O. Delalleau, N. L. Roux, and M. Ouimet. Out-of-sample extensions for LLE, isomap, MDS, eigenmaps, and spectral clustering. In *Advances in Neural Information Processing Systems 16*, 2003.

[6] M. Brand. Continuous nonlinear dimensionality reduction by kernel eigenmaps. In *International Joint Conference on Artificial Intelligence*, 2003.

[7] F. R. K. Chung. *Spectral Graph Theory*, volume 92 of *Regional Conference Series in Mathematics*. AMS, 1997.

[8] S. C. Deerwester, S. T. Dumais, T. K. Landauer, G. W. Furnas, and R. A. harshman. Indexing by latent semantic analysis. *Journal of the American Society of Information Science*, 41(6):391–407, 1990.

[9] J. H. Friedman. Regularized discriminant analysis. *Journal of the American Statistical Association*, 84(405):165–175, 1989.

[10] K. Fukunaga. *Introduction to Statistical Pattern Recognition*. Academic Press, 2nd edition, 1990.

[11] G. H. Golub and C. F. V. Loan. *Matrix computations*. Johns Hopkins University Press, 3rd edition, 1996.

[12] S. Guattery and G. L. Miller. Graph embeddings and laplacian eigenvalues. *SIAM Journal on Matrix Analysis and Applications*, 21(3):703–723, 2000.

[13] J. Ham, D. Lee, S. Mika, and B. Schölkopf. A kernel view of the dimensionality reduction of manifolds. 2004.

[14] T. Hastie, A. Buja, and R. Tibshirani. Penalized discriminant analysis. *Annals of Statistics*, 23:73–102, 1995.

[15] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. New York: Springer-Verlag, 2001.

[16] X. He, D. Cai, S. Yan, and H.-J. Zhang. Neighborhood preserving embedding. In *Proc. Int. Conf. Computer Vision (ICCV'05)*, 2005.

[17] X. He and P. Niyogi. Locality preserving projections. In *Advances in Neural Information Processing Systems 16*. MIT Press, Cambridge, MA, 2003.

[18] X. He, S. Yan, Y. Hu, P. Niyogi, and H.-J. Zhang. Face recognition using laplacianfaces. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(3):328–340, 2005.

[19] K. Lang. Newsweeder: Learning to filter netnews. In *Proceedings of the Twelfth International Conference on Machine Learning*, pages 331–339, 1995.

[20] K. V. Mardia, J. T. Kent, and J. M. Bibby. *Multivariate Analysis*. Academic Press, 1980.

[21] C. A. Micchelli. Algebraic aspects of interpolation. In *Proceedings of Symposia in Applied Mathematics*, volume 36, pages 81–102, 1986.

[22] A. Y. Ng, M. Jordan, and Y. Weiss. On spectral clustering: Analysis and an algorithm. In *Advances in Neural Information Processing Systems 14*, pages 849–856. MIT Press, Cambridge, MA, 2001.

[23] C. C. Paige and M. A. Saunders. Algorithm 583 LSQR: Sparse linear equations and least squares problems. *ACM Transactions on Mathematical Software*, 8(2):195–209, June 1982.

[24] C. C. Paige and M. A. Saunders. LSQR: An algorithm for sparse linear equations and sparse least squares. *ACM Transactions on Mathematical Software*, 8(1):43–71, March 1982.

[25] R. Penrose. A generalized inverse for matrices. In *Proceedings of the Cambridge Philosophical Society*, volume 51, pages 406–413, 1955.

[26] S. Roweis and L. Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290(5500):2323–2326, 2000.

[27] B. Schölkopf and A. J. Smola. *Learning with Kernels*. MIT Press, 2002.

[28] G. W. Stewart. *Matrix Algorithms Volume I: Basic Decompositions*. SIAM, 1998.

[29] G. W. Stewart. *Matrix Algorithms Volume II: Eigensystems*. SIAM, 2001.

[30] J. Tenenbaum, V. de Silva, and J. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500):2319–2323, 2000.

[31] M. Turk and A. Pentland. Eigenfaces for recognition. *Journal of Cognitive Neuroscience*, 3(1):71–86, 1991.

[32] Y. Weiss. Segmentation using eigenvectors: A unifying view. In *Proc. Int. Conf. Computer Vision (ICCV'99)*, pages 975–982, Kerkyra, Greece, Sept. 1999.

[33] S. Yan, D. Xu, B. Zhang, H.-J. Zhang, Q. Yang, and S. Lin. Graph embedding and extension: A general framework for dimensionality reduction. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(1), 2007.

[34] D. Zhou, O. Bousquet, T. Lal, J. Weston, and B. Schölkopf. Learning with local and global consistency. In *Advances in Neural Information Processing Systems 16*, 2003.

[35] X. Zhu, Z. Ghahramani, and J. Lafferty. Semi-supervised learning using gaussian fields and harmonic functions. In *Proc. of the twentieth Internation Conference on Machine Learning*, 2003.