Spectral Regression: A Unified Approach for Sparse Subspace Learning*

Deng Cai
UIUC
dengcai2@cs.uiuc.edu

Xiaofei He
Yahoo!
hex@yahoo-inc.com

Jiawei Han
UIUC
hanj@cs.uiuc.edu

Abstract

Recently the problem of dimensionality reduction (or, subspace learning) has received a lot of interests in many fields of information processing, including data mining, information retrieval, and pattern recognition. Some popular methods include Principal Component Analysis (PCA), Linear Discriminant Analysis (LDA) and Locality Preserving Projection (LPP). However, a disadvantage of all these approaches is that the learned projective functions are linear combinations of all the original features, thus it is often difficult to interpret the results. In this paper, we propose a novel dimensionality reduction framework, called **Unified Sparse Subspace Learning** *(USSL), for learning sparse projections. USSL casts the problem of learning the projective functions into a regression framework, which facilitates the use of different kinds of regularizers. By using a L_1-norm regularizer (*lasso*), the sparse projections can be efficiently computed. Experimental results on real world classification and clustering problems demonstrate the effectiveness of our method.*

1. Introduction

Dimensionality reduction has been a key problem in many fields of information processing, such as data mining, information retrieval, and pattern recognition. When data is represented as points in a high-dimensional space, one is often confronted with tasks like nearest neighbor search. Many methods have been proposed to index the data for fast query response, such as K-D tree, R tree, R* tree, etc [13]. However, these methods can only operate with small dimensionality, typically less than 100. The effectiveness and efficiency of these methods drop exponentially as the dimensionality increases, which is commonly referred to as the "curse of dimensionality".

To deal with this problem, the dimensionality reduction technique can be used. 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 [19]. Various researchers (see [2, 24, 26]) have considered the case when the data lives on or close to a low dimensional sub-manifold of the high dimensional ambient space. One hopes then to estimate geometrical and topological properties of the sub-manifold from random points ("scattered data") lying on this unknown sub-manifold. Along this direction, many subspace learning algorithms have been proposed for face recognition. Some popular ones include Locality Preserving Projection (LPP) [19], Neighborhood Preserving Embedding (NPE) [17] and Isometric Projection (IsoP) [5]. Despite the different motivations of these algorithms, they can be nicely interpreted in a general graph embedding framework [3, 19, 28].

One of the major disadvantages of all the above algorithms is that the learned projective functions are linear combinations of all the original features, thus it is often difficult to interpret the results. Recently, there are considerable interests on developing sparse subspace learning algorithms. Zou *et al*. [30] proposed an elegant sparse PCA algorithm (SPCA) using their "Elastic Net" framework for L_1-penalized regression on regular principle components, solved very efficiently using *least angle regression*

*The work was supported in part by the U.S. National Science Foundation NSF IIS-05-13678, NSF BDI-05-15813 and MIAS (a DHS Institute of Discrete Science Center for Multimodal Information Access and Synthesis). 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.

(LARS) [11]. Subsequently, d'Aspremont *et al*. [9] relaxed the hard cardinality constraint and solved for a convex approximation using semi-definite programming. In [21, 22], Moghaddam *et al*. proposed a spectral bounds framework for sparse subspace learning. Particularly, they proposed both exact and greedy algorithms for sparse PCA and sparse LDA.

In this paper, we propose a novel Unified Sparse Subspace Learning framework (USSL), for sparse projections learning. The proposed approach is fundamentally based on regression and spectral graph analysis [8]. Specifically, USSL decomposes the subspace learning as a two-step approach: graph embedding for responses learning and regression for projective functions learning. This decomposition links subspace leaning and regression. By incorporating the regression as a building block, different kinds of regularizers can be naturally incorporated. With a L_1-norm regularizer (*lasso* or *elastic net*), the sparse projections can be efficiently computed in USSL.

The specific contributions of this paper include:

- It reviews and provides a unified graph embedding analysis of many existing subspace learning algorithms, *e.g*., LDA, LPP and NPE (Section 2).
- It gives the formulation of sparse subspace learning and discusses the advantages and disadvantages of existing sparse subspace learning approaches (Section 3).
- It proposes a novel unified sparse subspace learning framework. This framework builds the connection between regression and many popular graph-based subspace learning algorithms. Their sparse solutions can be efficiently computed with a L_1-norm regularizer in the proposed framework (Section 4).
- We have performed extensive experimental comparisons on both supervised and unsupervised learning, which demonstrate the effectiveness of our method. (Section 5).

We summarize our findings and discuss extensions to the current work in Section 6, which concludes the paper.

2. Graph Embedding View of Subspace Learning

In this Section, we provide a general framework of analysis for the existing subspace learning algorithms from the graph embedding viewpoint.

Suppose we have m data samples $\{\mathbf{x}_i\}_{i=1}^m \subset \mathbb{R}^n$, $X = [\mathbf{x}_1, \cdots, \mathbf{x}_m]$. In the past decades, many dimensionality reduction algorithms have been proposed to find a low dimensional representation of $\mathbf{x}_i$. Despite the different motivations of these algorithms, they can be nicely interpreted in a general *graph embedding* framework [3, 19, 28].

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 the 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 $\mathbf{y}$ is given by minimizing

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

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 [15]. With some simple algebraic formulations, we have

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

where $L = D - W$ is the *graph Laplacian* [8] 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 a quadratically-constrained quadratic program (QCQP):

$$\begin{array}{rl} \textbf{min} & \mathbf{y}^T L \mathbf{y} \\ \text{subject to} & \mathbf{y}^T D \mathbf{y} = 1 \end{array}$$

The constraint $\mathbf{y}^T D\mathbf{y} = 1$ removes an arbitrary scaling factor in the embedding. Notice that $L = D - W$, it is easy to see that the above optimization problem has the following equivalent variation:

$$\begin{array}{rl} \textbf{max} & \mathbf{y}^T W \mathbf{y} \\ \text{subject to} & \mathbf{y}^T D \mathbf{y} = 1 \end{array} \tag{1}$$

The optimal $\mathbf{y}$'s can be obtained by solving the maximum eigenvalue eigen-problem [12]:

$$W\mathbf{y} = \lambda D\mathbf{y}. \tag{2}$$

Many recently proposed manifold learning algorithms, like ISOAMP [26], Laplacian Eigenmap [2], Locally Linear Embedding [24], can be interpreted in this framework with different choice of W.

The graph embedding approach described above only provides the mappings for the graph vertices in the training set. For classification purpose (*e.g*., face recognition, text categorization), a mapping for all samples, including new test samples, is required. If we choose a linear function, *i.e*., $y_i = f(\mathbf{x}_i) = \mathbf{a}^T\mathbf{x}_i$, we have $\mathbf{y} = X^T\mathbf{a}$. Eq. (1)

can be rewritten as:

$$\begin{array}{rl} \textbf{max} & \mathbf{a}^T X W X^T \mathbf{a} \\ \text{subject to} & \mathbf{a}^T X D X^T \mathbf{a} = 1 \end{array} \tag{3}$$

The optimal $\mathbf{a}$'s are the eigenvectors corresponding to the maximum eigenvalue of eigen-problem:

$$XWX^T\mathbf{a} = \lambda XDX^T\mathbf{a}. \tag{4}$$

This approach is called *Linear extension of Graph Embedding* (LGE). With different choices of W, the LGE framework leads to many popular linear dimensionality reduction algorithms, *e.g.*, LDA, LPP and NPE. We will briefly list the choices of W for these algorithms as follows.

LDA:

Suppose we have c classes and the t-th class have m_t samples, $m_1 + \cdots + m_c = m$. Define

$$W_{ij} = \begin{cases} 1/m_t, & \text{if } \mathbf{x}_i \text{ and } \mathbf{x}_j \text{ both belong to} \\ & \text{the } t\text{-th class;} \\ 0, & \text{otherwise.} \end{cases} \tag{5}$$

With such W, it is easy to check that $D = I$. Please see [19], [7] for the detailed derivation.

LPP:

Let $N_k(\mathbf{x}_i)$ denote the set of k nearest neighbors of $\mathbf{x}_i$.

$$W_{ij} = \begin{cases} e^{-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma^2}}, & \text{if } \mathbf{x}_i \in N_k(\mathbf{x}_j) \text{ or } \mathbf{x}_j \in N_k(\mathbf{x}_i) \\ 0, & \text{otherwise.} \end{cases} \tag{6}$$

For supervised case, one can also integrate the label information into W by searching the k nearest neighbors of $\mathbf{x}_i$ among the points sharing the same label with $\mathbf{x}_i$. Please see [19] for the details.

NPE:

Let $N_k(\mathbf{x}_i)$ denote the set of k nearest neighbors of $\mathbf{x}_i$ and M be a $m \times m$ local reconstruction coefficient matrix. M is defined as follows:

For i-th row of M, $M_{ij} = 0$ if $\mathbf{x}_j \notin N_k(\mathbf{x}_i)$. The other M_{ij} can be computed by minimizing the following objective function,

$$\min \|\mathbf{x}_i - \sum_{j \in N_k(\mathbf{x}_i)} M_{ij}\mathbf{x}_j\|^2, \quad \sum_{j \in N_k(\mathbf{x}_i)} M_{ij} = 1.$$

Define

$$W = M + M^T - M^T M \tag{7}$$

and it is easy to check that $D = I$. Please see [17], [28] for the detailed derivation.

All the above mentioned linear subspace learning algorithms need to solve the eigen-problem in Eqn. (4). To get a stable solution of this eigen-problem, the matrices XDX^T is required to be non-singular [14] which is not true when the number of features is larger than the number of samples. A popular way to deal with the singularity of XDX^T is to apply the idea of regularization, by adding some constant values to the diagonal elements of XDX^T, as $XDX^T + \alpha I$, for any $\alpha > 0$. It is easy to see that $XDX^T + \alpha I$ is nonsingular. The computational complexity of this approach scales as $O(n^3 + mn^2)$ where m is the number of samples and n is the number of features.

3. Sparse Subspace Learning Formulation

For simplicity, we define $A = XWX^T$, $B = XDX^T$ and rewrite the optimization problem of LGE in Eqn. (3) as:

$$\begin{array}{rl} \textbf{max} & \mathbf{a}^T A \mathbf{a} \\ \text{subject to} & \mathbf{a}^T B \mathbf{a} = 1 \end{array}$$

Following [22], we define the Sparse Subspace Learning (SSL) optimization in terms of the following cardinality-constrained QCQP:

$$\begin{array}{rl} \textbf{max} & \mathbf{a}^T A \mathbf{a} \\ \text{subject to} & \mathbf{a}^T B \mathbf{a} = 1 \\ & \text{card}(\mathbf{a}) = k \end{array} \tag{8}$$

The feasible set is all sparse $\mathbf{a} \in \mathbb{R}^n$ with k non-zero elements and $\text{card}(\mathbf{a})$ as their L_0-norm. Unfortunately, this optimization problem is NP-hard and therefor generally intractable .

In [21, 22], Moghaddam *et al*. proposed a spectral bounds framework for sparse subspace learning. Particularly, they proposed both exact and greedy algorithms for sparse PCA and sparse LDA. Their spectral bounds framework is based on the following optimal condition of the sparse solution.

A sparse vector $\mathbf{a} \in \mathbb{R}^n$ with cardinality k yielding the maximum objective value in Eqn. (8) would necessarily imply that

$$\lambda_{max} = \frac{\mathbf{a}^T A \mathbf{a}}{\mathbf{a}^T B \mathbf{a}} = \frac{\mathbf{b}^T A_k \mathbf{b}}{\mathbf{b}^T B_k \mathbf{b}}$$

where $\mathbf{b} \in \mathbb{R}^k$ contains the k non-zero elements in $\mathbf{a}$ and the $k \times k$ principle sub-matrices of A and B obtained by deleting the rows and columns corresponding to the zero indices of $\mathbf{a}$. The k-dimensional quadratic form in $\mathbf{b}$ is equivalent to a standard unconstrained generalized Rayleigh quotient, which can be solved by a generalized eigen-problem.

The above observation gives the exact algorithm for sparse subspace learning: a discrete search for the k indices which maximize λ_{max} of the subproblem (A_k, B_k). However, such observation does not suggest an efficient algorithm because an exhaustive search is still NP-hard. To solve this problem, Moghaddam *et al*. proposed an efficient greedy algorithm which combines *backward elimination* and *forward selection* [21, 22]. As we discussed in

Section 2, many of the popular graph-based subspace learning algorithms can be formulated as the generalized eigen-problem, Moghaddam's approach provides a general solution for learning sparse projections in all these subspace learning algorithms. However, there are two major drawbacks of their approach:

1. Even their algorithm is a greedy one, the cost of backward elimination is with complexity $O(n^4+mn^2)$[22].
2. In reality, more than one projective functions are usually necessary for subspace learning. However, the optimal condition of the sparse solution only gives the guide to find ONE sparse "eigenvector", which is the first projective function. It is unclear how to find the following projective functions. Although [21] suggests to use recursive deflation, the sparseness of the the following projective functions is not guaranteed.

In [30], Zou *et al*. proposed an elegant sparse PCA algorithm (SPCA) using their "Elastic Net" framework for L_1-penalized regression on regular principle components, solved very efficiently using *least angle regression* (LARS) [11]. The key idea of SPCA is formulating PCA as a regression-type optimization problem.

Without loss of generality, we assume the data are centered[1]. The PCA objective function is

$$\begin{array}{rl} \textbf{max} & \mathbf{a}^T X X^T \mathbf{a} \\ \text{subject to} & \mathbf{a}^T \mathbf{a} = 1 \end{array} \tag{9}$$

and the optimal $\mathbf{a}$'s are the eigenvectors with respect to the maximum eigenvalues of the following eigen-problem:

$$XX^T\mathbf{a} = \lambda\mathbf{a}. \tag{10}$$

Suppose the rank of X is r and the Singular Value Decomposition (SVD) of X is:

$$X = U\Sigma V^T, \tag{11}$$

it is easy to verify that the column vectors in U are the eigenvectors of XX^T [14], *i.e*., the projective functions of PCA. Let $Y = [\mathbf{y}_1, \cdots, \mathbf{y}_r] = U^T X = \Sigma V^T$, each row vector of Y is the sample vector in the r-dimensional PCA subspace. Thus, the projective functions of PCA are essentially the solutions of the linear equation systems:

$$X^T\mathbf{a}_t = \mathbf{y}_t, \quad t = 1, \cdots, r$$

in other words, $\mathbf{a}_t$ is the solution of the regression system:

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

[1]This can be achieved by subtracting the mean vector from all the sample vectors.

where y_i^t is the i-th element of $\mathbf{y}_t$. Zou *et al*. [30] add L_1-regularizer to get the sparse solutions:

$$\mathbf{a}_t = \arg\min_{\mathbf{a}} \sum_{i=1}^{m} (\mathbf{a}^T\mathbf{x}_i - y_i^t)^2 + \beta \sum_{j=1}^{n} |a_j|$$

where a_j is the j-th element of $\mathbf{a}$. The above regression problem is called *Lasso* [16] and can be efficiently computed using LARS algorithm [11].

If we want to apply the similar technique to those linear graph embedding algorithms (LGE), the key problem is how we can formulate LGE as a regression-type optimization problem.

4. Unified Sparse Subspace Learning via Regression

In this section, we describe our regression formulation of graph based subspace learning which is the key of the proposed unified sparse subspace learning approach.

4.1. Spectral Regression

In order to formulate LGE as a regression-type optimization problem, we use the following theorem:

Theorem 1 *Let* $\boldsymbol{y}$ *be the eigenvector of eigen-problem in Eqn. (2) with eigenvalue* λ*. If* $X^T\boldsymbol{a} = \boldsymbol{y}$*, then* $\boldsymbol{a}$ *is the eigenvector of eigen-problem in Eqn. (4) with the same eigenvalue* λ*.*

Proof We have $W\mathbf{y} = \lambda D\mathbf{y}$. At the left side of Eqn. (4), 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. (4) with the same eigenvalue λ. ∎

Theorem (1) shows that instead of solving the eigen-problem in Eqn. (4), the linear projective functions can be obtained through two steps:

1. Solve the eigen-problem in Eqn. (2) 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:

$$\mathbf{a} = \arg\min_{\mathbf{a}} \sum_{i=1}^{m} (\mathbf{a}^T\mathbf{x}_i - y_i)^2 \tag{12}$$

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

It is clear that Eqn. (12) is exactly what we want, the regression-type formulation of LGE problem.

In the situation that the number of samples is smaller than the number of features, the minimization problem (12) is *ill posed*. We may have infinitely many solutions to the linear equations system $X^T\mathbf{a} = \mathbf{y}$ (the system is underdetermined). The most popular way to solve this problem is to apply the regularization technique, *i.e.*, impose a penalty on the norm of $\mathbf{a}$. L_2-norm and L_1-norm are two of the most popular ones.

With a L_2-norm on $\mathbf{a}$, we have

$$\mathbf{a} = \arg\min_{\mathbf{a}} \left(\sum_{i=1}^{m} \left(\mathbf{a}^T \mathbf{x}_i - y_i \right)^2 + \alpha \sum_{j=1}^{n} a_j^2 \right), \quad (13)$$

where a_j is the j-th element of $\mathbf{a}$. This is usually referred as *ridge regression* in statistics [16]. The $\alpha \geq 0$ is a parameter to control the amounts of shrinkage. The ridge penalty does not provide a sparse solution.

With a L_1-norm on $\mathbf{a}$, we have

$$\mathbf{a} = \arg\min_{\mathbf{a}} \left(\sum_{i=1}^{m} \left(\mathbf{a}^T \mathbf{x}_i - y_i \right)^2 + \beta \sum_{j=1}^{n} |a_j| \right), \quad (14)$$

which is usually referred as *lasso regression* [16]. Due to the nature of the L_1 penalty, some coefficients will be shrunk to exact zero if β is large enough. Therefore the lasso produces a sparse model, which is exactly what we want. However, the lasso has several limitations as pointed out in [29]. The most relevant one to this work is that the number of selected features by the lasso is limited by the number of samples. For example, if applied to the face image data where there are thousands of features ($n > 1000$) with less than 100 samples ($m < 100$), the lasso can only select at most m features, which is clearly unsatisfactory. The Elastic Net [29] generalizes the lasso to overcome its drawbacks by combining both the ridge and lasso penalty:

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

When $\alpha > 0$ or $\beta > 0$, the solution of the above optimization problem will not satisfy the linear equations system $X^T\mathbf{a} = \mathbf{y}$ and $\mathbf{a}$ will not be the eigenvector of eigen-problem in Eqn. (4). It is interesting and important to see when the solution of the optimization problem in Eqn. (15) gives the exact solutions of eigen-problem (4). Specifically, we have the following theorem:

Theorem 2 *Suppose* $\mathbf{y}$ *is the eigenvector of eigen-problem in Eqn. (2), if* $\mathbf{y}$ *is in the space spanned by row vectors of* X*, the solution of the optimization problem in Eqn. (15) will be the eigenvector of eigen-problem in Eqn. (4) as* α *and* β *decease to zero.*

Proof See Appendix A. ∎

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 which is shown in the following Corollary.

Corollary 3 *If the sample vectors are linearly independent,* i.e.*,* $rank(X) = m$*, all the solution of the optimization problem in Eqn. (15) (with different eigenvector* $\mathbf{y}$*'s) are the eigenvectors of eigen-problem in Eqn. (4) as* α *and* β *deceases to zero.*

Proof See Appendix B. ∎

Our above two-step approach essentially performs regression after the spectral analysis of the graph, we called it *Spectral Regression* (SR) [6]. With lasso penalty, it provides a Unified Sparse Subspace Learning framework (USSL).

4.2. Eigenvectors of Eigen-problem in Eqn. (2)

With different choices of W, the optimization problem in Eqn. (3) gives the solutions of various subspace learning algorithms, *i.e.*, LDA, LPP and NPE. Thus, the USSL approach introduced in the previous section provides the sparse solutions of LDA, LPP and NPE.

Generally, we need to solve the eigen-problem in Eqn. (2) to get the responses vectors $\mathbf{y}$'s. In some cases, *i.e.* LDA, the W has a block diagonal structure and there is no need to solve the eigen-problem.

Without loss of generality, we assume that the data points in $\{\mathbf{x}_1, \cdots, \mathbf{x}_m\}$ are ordered according to their labels. It is easy to check that the matrix W defined in Eqn. (5) has a block-diagonal structure

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

where c is the number of classes, m_t is the number of samples in t-th class and $\{W^{(t)}\}_{t=1}^{c}$ is a $m_t \times m_t$ matrix with all the elements equal to $1/m_t$. Since the W is block-diagonal, its eigenvalues and eigenvectors are the union of the eigenvalues and eigenvectors of its blocks (the latter padded appropriately with zeros) [14]. It is straightforward to show that $W^{(t)}$ has eigenvector $\mathbf{e}^{(t)} \in \mathbb{R}^{m_t}$ associated with eigenvalue 1, where $\mathbf{e}^{(t)} = [1, 1, \cdots, 1]^T$. Also there is only one non-zero eigenvalue of $W^{(t)}$ because the rank of $W^{(t)}$ is 1. Thus, there are exactly c eigenvectors of W with the same eigenvalue 1. These eigenvectors are

$$\mathbf{y}_t = [\underbrace{0, \cdots, 0}_{\sum_{i=1}^{t-1} m_i}, \underbrace{1, \cdots, 1}_{m_t}, \underbrace{0, \cdots, 0}_{\sum_{i=t+1}^{c} m_i}]^T. \quad (16)$$

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 c eigenvectors. The vector of all ones $\mathbf{e}$ 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 $\mathbf{e}$ 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.

In binary classification case, the above procedure will produce one response vector

$$\mathbf{y} = [\underbrace{\frac{m}{m_1}, \cdots, \frac{m}{m_1}}_{m_1}, \underbrace{\frac{-m}{m_2}, \cdots, \frac{-m}{m_2}}_{m_2}]^T. \quad (17)$$

This is consistent with the previous well-known result on the relationship between LDA and regression for a binary problem [16]. The framework proposed in this paper extends this relation to multi-class case. Moreover, this framework also establishes the connection between regression and many other graph based subspace learning algorithms, *e.g.*, LPP, NPE.

4.3. Computational Complexity of USSL

The USSL computation involves two steps: responses generation (calculate the eigenvectors of eigen-problem in Eqn. (2)) and regularized regression.

For the W in LDA, the cost of the first step is mainly the cost of Gram-Schmidt method, which is $O(mc^2)$ [25]. For a k-NN graph W in LPP, the cost of the first step is $O(m^2n + m^2 \log m + qdmk)$. $O(m^2n)$ is used to calculate the pairwise distance between m samples with n features and $O(m^2 \log m)$ is used for k-nearest neighbors finding for all the m samples. The k-NN graph matrix W is sparse and the Lanczos algorithm [14] can be used to efficiently compute the first d eigenvectors of the eigen-problem in Eqn. (2) within $O(qdmk)$, where q is number of iterations in Lanczos.

All of the three types of regularized regression problems can be solved in $O(n^3 + mn^2)$ [16][11]. By using the *Least Angel Regression* (LARS) algorithm [11], the entire solution path (the solutions with all the possible cardinality on $\mathbf{a}$) of lasso and elastic net with a specific α can be computed in $O(n^3 + mn^2)$.

Considering $m \gg c$ and $m \gg d$, USSL provides a sparse LDA solution with $O(n^3 + mn^2)$ complexity and a sparse LPP solution with $O(m^2n + m^2 \log m + n^3 + mn^2)$ complexity. This complexity is exactly the same as the ordinary non-sparse solution solved by generalized eigen-problem. Comparing to the $O(n^4 + mn^2)$ greedy algorithm described in [22], USSL is much more efficient.

5. Experimental Results

In this section, we investigate the performance of our proposed USSL approach for both supervised learning (face recognition) and unsupervised learning (face clustering). All of our experiments have been performed on an Intel Pentium D 3.20GHz Linux machine with 2GB memory.

Two face databases were used in the experiment. The first one is the PIE (Pose, Illumination, and Experience) database[2] from CMU, and the second one is the Extended Yale-B database[3].

The CMU PIE face database contains 68 human subjects with 41,368 face images as a whole. The face images were captured by 13 synchronized cameras and 21 flashes, under varying pose, illumination and expression. We choose the frontal poses (C27) and use all the images under different illuminations and expressions, thus we get 3329 face images in total.

The Extended Yale-B face database contains 16128 images of 38 human subjects under 9 poses and 64 illumination conditions. In this experiment, we choose the frontal pose and use all the images under different illumination. Finally we get 2414 images in total.

All the face images are manually aligned and cropped. The size of each cropped image is 32×32 pixels, with 256 gray levels per pixel. Thus each image is represented as a 1024-dimensional vector.

5.1. USSL for Supervised Learning

In this experiment, we use the W in Eqn. (5). Thus, USSL provides a sparse LDA solution. We compare our algorithm with PCA, LDA and SparsePCA [30]. In face recognition, PCA and LDA are also called Eigenface [27] and Fisherface [1]. They are two of the most popular linear methods for face recognition. We do not compare with Sparse LDA [22] since it can only be applied to two-class case. Please refer to [22] for the details.

For each database, r $(= 33, 50, 67)$ percent of samples are randomly selected for training and the rest are used for testing. The training samples are used to learn the basis functions. By using these basis functions, the testing images can be mapped into lower dimensional subspace where recognition is carried out by using nearest neighbor classifier. 5-fold cross validation has been performed in SparsePCA and USSL for selecting the best cardinality of the basis functions. The choices of the cardinality are 10, 20, $\cdots$ 100, 150, 200, $\cdots$, 1000, 1024.

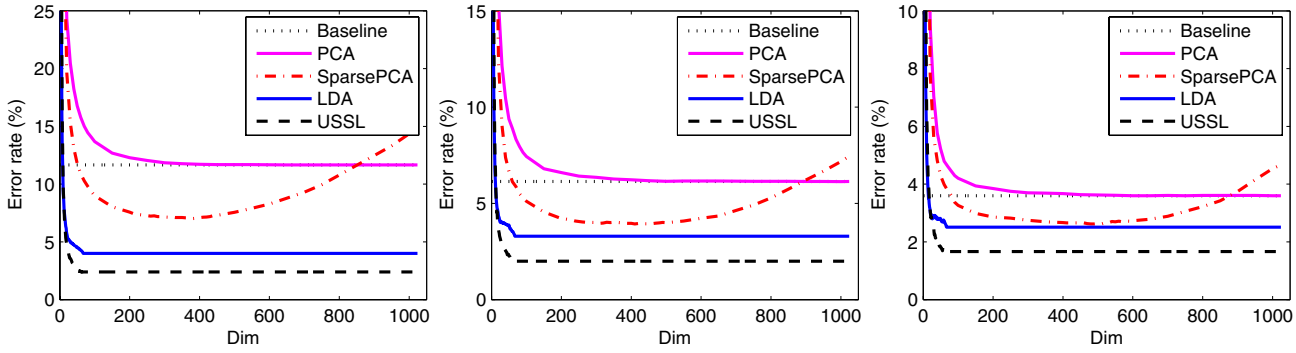
For each given r, we average the recognition results over 20 random splits. Figure 1 and 2 show the plots of error rate versus dimensionality reduction for the PCA, SparsePCA, LDA, USSL and baseline methods on PIE and

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

[3] http://vision.ucsd.edu/∼leekc/ExtYaleDatabase/ExtYaleB.html

Table 1. Comparison of classification error rate on PIE

Method	33% Training			50% Training			67% Training		
	error (%)	dim	sparsity	error (%)	dim	sparsity	error (%)	dim	sparsity
Baseline	11.7±0.5	1024	–	6.1±0.7	1024	–	3.6±0.6	1024	–
PCA	11.7±0.5	700	0	6.1±0.7	1000	0	3.6±0.6	1000	0
SparsePCA	7.0±0.6	380	92.2%	3.9±0.5	410	92.2%	2.6±0.5	480	92.2%
LDA	4.0±0.2	67	0	3.3±0.3	67	0	2.5±0.5	67	0
USSL	2.4±0.2	64	90.2%	2.0±0.2	67	90.2%	1.6±0.3	66	90.2%

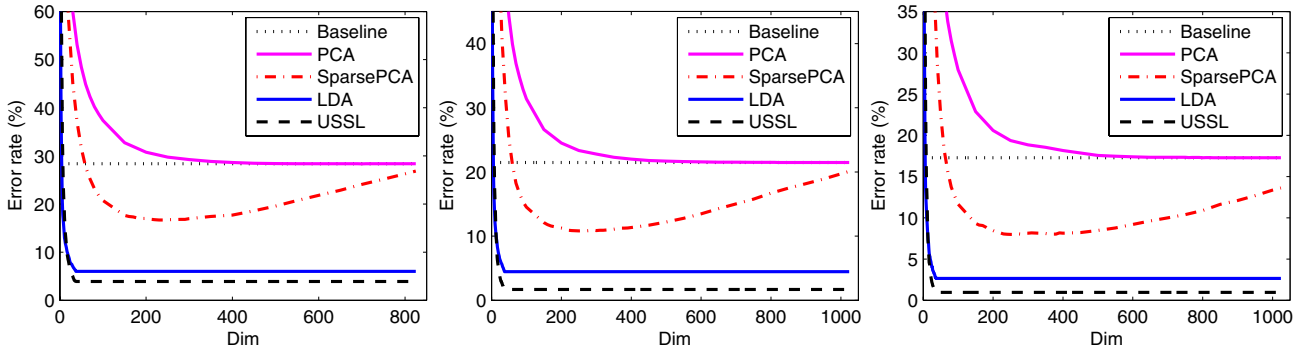


(a) 33% Training (b) 50% Training (c) 67% Training

Figure 1. Error rate vs. dimensionality reduction on PIE database

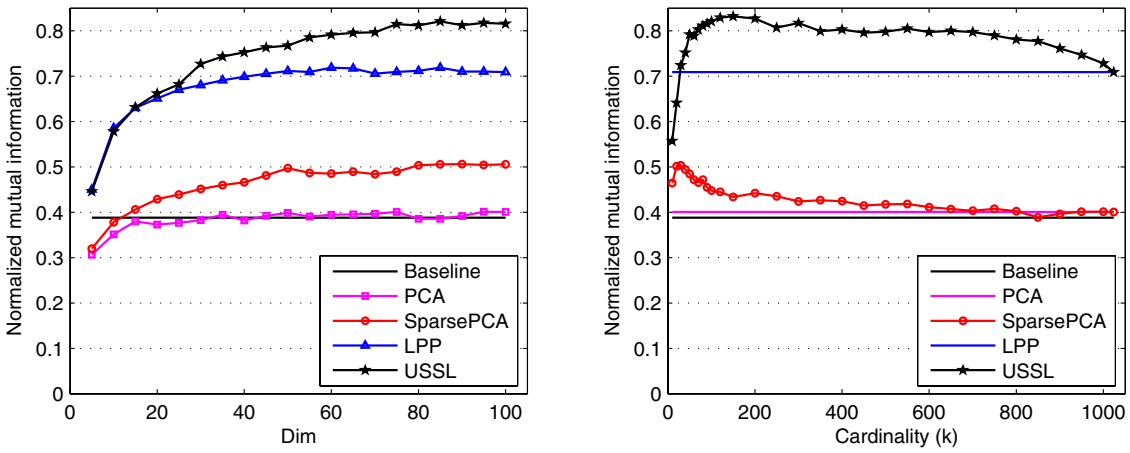
Table 2. Comparison of classification error rate on Yale-B

Method	33% Training			50% Training			67% Training		
	error (%)	dim	sparsity	error (%)	dim	sparsity	error (%)	dim	sparsity
Baseline	28.4±1.3	1024	–	21.5±1.3	1024	–	17.3±0.7	1024	–
PCA	28.4±1.3	700	0	21.5±1.3	860	0	17.3±0.7	830	0
SparsePCA	16.7±1.1	230	95.1%	10.7±0.9	270	95.1%	8.0±0.5	250	95.1%
LDA	6.0±0.6	37	0	4.5±0.5	37	0	2.7±0.5	37	0
USSL	3.9±0.6	37	86.3%	1.7±0.4	37	86.3%	1.0±0.3	37	86.3%



(a) 33% Training (b) 50% Training (c) 67% Training

Figure 2. Error rate vs. dimensionality reduction on Yale-B database



(a) (b)

Figure 3. Normalized mutual information vs. dimensionality (a) and Normalized mutual information vs. cardinality (b) on PIE database

Yale-B databases, respectively. For the baseline method, the recognition is simply performed in the original 1024-dimensional image space without any dimensionality reduction. Note that, the upper bound of the dimensionality of LDA is $c-1$ where c is the number of individuals [10]. We use the LDA graph W as defined in Section 2 in our USSL algorithm. Thus, the upper bound of the dimensionality of USSL is also $c-1$. As can be seen, the performance of the PCA, SparsePCA, LDA and USSL algorithms varies with the number of dimensions. We show the best results together with the standard deviations obtained by them in Table 1 and 2 and the corresponding face subspaces are called optimal face subspace for each method. Particularly, we also shown the sparsity of the basis functions for these algorithms. The sparsity is computed as the ratio of the number of zero entries and the total number of entries. As can be seen, the sparsity for PCA and LDA are both zero, while the sparsity for sparse PCA and USSL are very high.

5.2. USSL for Unsupervised Learning

In this subsection, we investigate the use of our proposed approach for face clustering. Face clustering is an unsupervised task and we compare our algorithm with PCA, SparsePCA and Locality Preserving Projection (LPP) [18][19]. We use the same p-nearest neighbor graph in LPP and USSL. Thus, USSL provides a sparse LPP solution. We empirically set the value of p to 5.

We choose K-means as our clustering algorithm. K-means can be performed in the original feature space (Baseline) or in the reduced feature space (by using the dimensionality reduction algorithms, *e.g.*, PCA, LPP and USSL). The clustering result is evaluated by comparing the obtained label of each image with that provided by the ground truth. We use the normalized mutual information ($\overline{MI}$) to measure the clustering performance [4]. Let C denote the set of clusters obtained from the ground truth and C' obtained from an algorithm. Their mutual information metric $MI(C,C')$ is defined as follows:

$$MI(C,C') = \sum_{c_i \in C, c'_j \in C'} p(c_i,c'_j) \cdot log_2 \frac{p(c_i,c'_j)}{p(c_i) \cdot p(c'_j)}$$

where $p(c_i)$ and $p(c'_j)$ are the probabilities that a sample arbitrarily selected from the data set belongs to the clusters c_i and c'_j, respectively, and $p(c_i,c'_j)$ is the joint probability that the arbitrarily selected document belongs to the clusters c_i as well as c'_j at the same time. In our experiments, we use the normalized mutual information $\overline{MI}$ as follows:

$$\overline{MI}(C,C') = \frac{MI(C,C')}{\max(H(C),H(C'))}$$

where $H(C)$ and $H(C')$ are the entropies of C and C', respectively. It is easy to check that $\overline{MI}(C,C')$ ranges from 0 to 1. $\overline{MI}=1$ if the two sets of clusters are identical, and $\overline{MI}=0$ if the two sets are independent.

Figure (3(a)) shows the plot of normalized mutual information versus dimensionality for the PCA, SparsePCA, LPP, USSL and baseline methods. As can be seen, all the methods obtain the best performance with dimensionality less than 100, and there is no performance improvement with more dimensions. Our USSL algorithm outperforms the other four methods. LPP performs the second best. PCA performs the worst, close to the baseline.

Figure (3(b)) shows the performances of all the algorithm in the 100-dimensional subspace. We show the performance change with the cardinality of basis functions in SparsePCA and USSL. As can be seen, the best performance is obtained with relatively small cardinality.

6. Conclusions

In this paper, we proposed a novel unified framework for learning sparse projections. Our framework is developed from a graph embedding viewpoint of dimensionality reduction algorithms. It combines the spectral graph analysis and regularized regression to provide an efficient and effective approach for sparse subspace learning problem. Many recently proposed linear subspace learning algorithms, *e.g.*, LDA [1], LPP [18], NPE [17] and IsoP [5] can be interpreted as the linear extensions of specific graph embedding. Thus, all these algorithms can be fit into our framework and get sparse solutions. Extensive experimental results show effectiveness of the proposed approach.

References

[1] 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.

[2] 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.

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

[4] D. Cai, X. He, and J. Han. Document clustering using locality preserving indexing. *IEEE Transactions on Knowledge and Data Engineering*, 17(12):1624–1637, December 2005.

[5] D. Cai, X. He, and J. Han. Isometric projection. In *Proc. 2007 AAAI Conf. on Artificial Intelligence (AAAI-07)*, 2007.

[6] D. Cai, X. He, and J. Han. Spectral regression for dimensionality reduction. Technical report, Computer Science Department, UIUC, UIUCDCS-R-2007-2856, May 2007.

[7] D. Cai, X. He, and J. Han. SRDA: An efficient algorithm for large scale discriminant analysis. Technical report, Computer Science Department, UIUC, UIUCDCS-R-2007-2857, May 2007.

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

[9] A. d'Aspremont, L. E. Chaoui, M. I. Jordan, and G. R. G. Lanckriet. A direct formulation for sparse PCA using semidefinite programming. In *Advances in Neural Information Processing Systems 17*, 2004.

[10] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification*. Wiley-Interscience, Hoboken, NJ, 2nd edition, 2000.

[11] B. Efron, T. Hastie, I. Johnstone, and R. Tibshirani. Least angle regression. *Annals of Statistics*, 32(2):407–499, 2004.

[12] R. Fletcher. *Practical methods of optimization*. John Wiley and Sons, 2nd edition edition, 1987.

[13] V. Gaede and O. Günther. Multidimensional access methods. *ACM Comput. Surv.*, 30(2):170–231, 1998.

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

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

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

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

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

[19] 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.

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

[21] B. Moghaddam, Y. Weiss, and S. Avidan. Spectral bounds for sparse PCA: Exact and greedy algorithms. In *Advances in Neural Information Processing Systems 18*, 2005.

[22] B. Moghaddam, Y. Weiss, and S. Avidan. Generalized spectral bounds for sparse LDA. In *ICML '06: Proceedings of the 23rd international conference on Machine learning*, pages 641–648, 2006.

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

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

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

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

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

[28] S. Yan, D. Xu, B. Zhang, and H.-J. Zhang. Graph embedding: A general framework for dimensionality reduction. In *Proc. 2005 Internal Conference on Computer Vision and Pattern Recognition*, 2005.

[29] H. Zhou and T. Hastie. Regression shrinkage and selection via the elastic net, with applications to microarrays. Technical report, Statistics Department, Stanford University, 2003.

[30] H. Zhou, T. Hastie, and R. Tibshirani. Sparse principle component analysis. Technical report, Statistics Department, Stanford University, 2004.

Appendix

A. Proof of Theorem 2

Proof Let $\beta = 0$, the regularized least squares in Eqn. (15) can be rewritten in the matrix form as:

$$\mathbf{a} = \arg\min_{\mathbf{a}} \left((X^T\mathbf{a} - \mathbf{y})^T (X^T\mathbf{a} - \mathbf{y}) + \alpha \mathbf{a}^T\mathbf{a} \right). \quad (18)$$

Requiring the derivative of right side with respect to $\mathbf{a}$ vanish, we get

$$\begin{aligned} & (XX^T + \alpha I)\mathbf{a} = X\mathbf{y} \\ \Rightarrow\ & \mathbf{a} = (XX^T + \alpha I)^{-1} X\mathbf{y} \end{aligned} \quad (19)$$

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

$$X = U\Sigma V^T$$

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:

$$\begin{aligned} & V\mathbf{b} = \mathbf{y} \\ \Rightarrow\ & V^TV\mathbf{b} = V^T\mathbf{y} \\ \Rightarrow\ & \mathbf{b} = V^T\mathbf{y} \\ \Rightarrow\ & VV^T\mathbf{y} = \mathbf{y} \end{aligned} \quad (20)$$

To continue our proof, we need introduce the concept of pseudo inverse of a matrix [23], 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

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

The above limit exists even if X^TX is singular and $(X^TX)^{-1}$ does not exist [23]. Thus, the regularized least squares solution in Eqn. (19)

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

Combine with the equation in Eqn. (20), we have

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

By Theorem (1), $\mathbf{a}$ is the eigenvector of eigen-problem in Eqn. (4). ∎

B. Proof of Corollary 3

Proof The matrices W and D are of size $m \times m$ and there are m eigenvectors $\{\mathbf{y}_j\}_{j=1}^m$ of eigen-problem (2). Since $rank(X) = m$, all these m eigenvectors $\mathbf{y}_j$ are in the space spanned by row vectors of X. By Theorem (2), all m corresponding $\mathbf{a}_j$ in Eqn (19) are eigenvectors of eigen-problem in Eqn. (4) as α and β decreases to zero. ∎