Orthogonal Neighborhood Preserving Projections: A projection-based dimensionality reduction technique *

E. Kokiopoulou † Y. Saad‡

March 21, 2006

Abstract

This paper considers the problem of dimensionality reduction by orthogonal projection techniques. The main feature of the proposed techniques is that they attempt to preserve both the intrinsic neighborhood geometry of the data samples and the global geometry. In particular we propose a method, named Orthogonal Neighborhood Preserving Projections, which works by first building an "affinity" graph for the data, in a way that is similar to the method of Locally Linear Embedding (LLE). However, in contrast with the standard LLE where the mapping between the input and the reduced spaces is implicit, ONPP employs an explicit linear mapping between the two. As a result, handling new data samples becomes straightforward, as this amounts to a simple linear transformation. We show how to define kernel variants of ONPP, as well as how to apply the method in a supervised setting. Numerical experiments are reported to illustrate the performance of ONPP and to compare it with a few competing methods.

1 Introduction

The problem of dimensionality reduction appears in many fields including data mining, machine learning and computer vision, to name just a few. The goal of dimensionality reduction is to map the high dimensional samples to a lower dimensional space such that certain properties are preserved. Usually, the property that is preserved is quantified by an objective function and the dimensionality reduction problem is formulated as an optimization problem. For instance, Principal Components Analysis (PCA) is a traditional linear technique which aims at preserving the global variance and relies on the solution of an eigenvalue problem involving the sample covariance matrix. Locally Linear Embedding (LLE) [7, 11] is a nonlinear dimensionality reduction technique which aims at preserving the local geometries at each neighborhood.

*Work supported by NSF under grant DMS 0510131 and by the Minnesota Supercomputing Institute.

†EPFL, LTS4 lab, Bat. ELE, Station 11; CH 1015 Lausanne; Switzerland. Email: efrosyni.kokiopoulou@epfl.ch

‡Department of Computer Science and Engineering; University of Minnesota; Minneapolis, MN 55455. Email: saad@cs.umn.edu.

While PCA is good at preserving the global structure, it does not preserve the locality of the data samples. In this paper, a *linear* dimensionality reduction technique is advocated, which preserves the intrinsic geometry of the local neighborhoods. The proposed method, named Orthogonal Neighborhood Preserving Projections (ONPP), projects the high dimensional data samples on a lower dimensional space by means of a linear transformation V. The dimensionality reduction matrix V is obtained by minimizing an objective function which captures the discrepancy of the intrinsic neighborhood geometries in the reduced space. Experimental evidence seems to indicate that this feature is crucial in preserving the global geometry as well, via the interaction of overlapping neighborhoods. In particular, this suggests that ONPP can be effectively used for data visualization purposes and that it may be viewed as a synthesis of PCA and LLE.

ONPP constructs a weighted k-nearest neighbor (k-NN) graph which models explicitly the data topology. Similarly to LLE, the weights are built to capture the geometry of the neighborhood of each point. The linear projection step is determined by imposing the constraint that each data sample in the reduced space is reconstructed from its neighbors by the same weights used in the input space. However, in contrast to LLE, ONPP computes an explicit linear mapping from the input space to the reduced space. Note that in LLE the mapping is implicit and it is not clear how to embed new data samples (see e.g. research efforts by Bengio et al. [2]). In the case of ONPP the projection of a new data sample is straightforward as it simply amounts to a matrix by vector product.

ONPP shares some properties with Locality Preserving Projections (LPP) [4]. Both are linear dimensionality reduction techniques which construct the k-NN graph in order to model the data topology. However, our algorithm uses the optimal data-driven weights of LLE which reflect the intrinsic geometry of the local neighborhoods, whereas the uniform weights (0/1) used in LPP aim at preserving locality without explicit consideration to the local geometric structure. Note that Gaussian weights can be used in LPP but these are somewhat artificial and require the selection of an appropriate value of the parameter σ, the width of the Gaussian envelope. Although this issue is often overlooked, it is crucial for the performance of the method and remains a serious handicap when using Gaussian weights. Experimental results suggest that ONPP is effective in conveying meaningful local and global geometric information from high dimensional samples to low dimensional ones. A second significant difference between LLE and ONPP, is that the latter forces the projection to be orthogonal. In LLE, the projection is defined via a certain objective function, whose minimization leads to eigenvectors of a generalized eigenvalue problem.

2 Dimensionality reduction by projection

Given a dataset $X = [x_1, x_2, \ldots, x_n] \in R^{m \times n}$ and the dimension d of the reduced space, with $d \ll m$, the goal of dimensionality reduction is to produce a set Y which is an accurate representation of X, but of smaller dimension. This can be achieved in different ways by selecting the *type* of the reduced dimension Y as well as the desirable *properties to be preserved*. By type we mean whether we require that Y be simply a low-rank representation of X, or a data set in a vector space with fewer dimensions. Examples of properties to be preserved include the global geometry, or neighborhood information.

Projection-based techniques consist of replacing the original data X by a matrix of the form

$$Y = V^\top X, \quad \text{where} \quad V \in R^{m\times d}. \tag{1}$$

Thus, each vector x_i is replaced by $y_i = V^\top x_i$ a member of the d-dimensional space R^d. If V is a unitary matrix, then Y represents the orthogonal projection of X into the V-space.

The best known technique in this category is Principal Component Analysis (PCA). PCA computes V such that the variance of the projected vectors is maximized, i.e, V is the maximizer of

$$\max_{\substack{V \in R^{m\times d} \\ V^\top V = I}} \left\| y_i - \frac{1}{n}\sum_{j=1}^{n} y_j \right\|_2^2, \quad y_i = V^\top x_i.$$

As can be easily shown, the matrix V which maximizes the above quantity is simply the set of left singular vectors of the matrix $X(I - \frac{1}{n}ee^\top)$, associated with the largest d singular values (e is the vector of ones).

2.1 LPP and OLPP

Another related techniques is that of *Locality Preserving Projections*. LPP projects the data so as to preserve a certain *affinity graph* constructed for the data. This graph can be defined for example by taking a certain nearness measure and include all points within a radius ϵ of a given vertex, to its adjacency list. Alternatively, one can include those k nodes that are the nearest neighbors to x_i. The weights can be defined in different ways as well. Two common choices are weights of the heat kernel $w_{ij} = exp(-\|x_i - x_j\|_2^2/t)$ or constant weights ($w_{ij} = 1$ if i and j are adjacent, $w_{ij} = 0$ otherwise). The adjacency graph along with these weights defines a matrix W whose entries are the weights w_{ij}'s which are nonzero only for adjacent nodes in the graph. Note that the entries of W are nonnegative and that W is sparse and symmetric.

LPP defines the projected points in the form $y_i = V^\top x_i$ by *putting a high penalty for mapping nearest neighbor nodes in the original graph to distant points in the projected data*. Specifically, the objective function to be minimized is

$$E_{lpp} = \frac{1}{2}\sum_{i,j=1}^{n} w_{ij}\|y_i - y_j\|_2^2 \tag{2}$$

Note that the matrix V is implicitly represented in the above function, through the dependence of the y_is on V.

The following theorem expresses the above objective function as a trace.

Theorem 2.1 *Let W be a certain symmetric affinity graph, and define $D = diag(d_i)$ with $d_i = \sum_{j=1}^{n} w_{ij}$. Let the points y_i be defined to be the columns of $Y = V^\top X$ where $V \in R^{m\times d}$. Then the objective function (2) is equal to*

$$E_{lpp} = tr[Y(D-W)Y^\top] = tr[V^\top X(D-W)X^\top V] \tag{3}$$

Proof. By definition:

$$\begin{aligned} E_{lpp} &= \frac{1}{2}\sum_{i,j=1}^{n} w_{ij}\|y_i - y_j\|_2^2 \\ &= \frac{1}{2}\sum_{i,j=1}^{n} w_{ij}(y_i - y_j)^\top (y_i - y_j) \\ &= \frac{1}{2}\sum_{i,j=1}^{n} w_{ij} y_i^\top y_i + \frac{1}{2}\sum_{i,j=1}^{n} w_{ij} y_j^\top y_j - \sum_{i,j=1}^{n} w_{ij} y_i^\top y_j \\ &= \sum_{i,j=1}^{n} w_{ij} y_i^\top y_i - \sum_{i,j=1}^{n} w_{ij} y_i^\top y_j \\ &= \sum_{i}^{n} d_i y_i^\top y_i - \sum_{i,j=1}^{n} w_{ij} y_i^\top y_j. \end{aligned}$$

An observation will simplify the first term of the above expression:

$$\sum_{i=1}^{n} d_i y_i^\top y_i = tr(DY^\top Y) = tr[YDY^\top]$$

and similarly, for the second term we have

$$\begin{aligned} \sum_{i,j=1}^{n} w_{ij} y_i^\top y_j &= \sum_{i}^{n} (Ye_i)^\top \sum_{j=1}^{n} w_{ji} y_j \\ &= \sum_{i}^{n} e_i^\top Y^\top (YW) e_i \\ &= tr[Y^\top (YW)] \\ &= tr[YWY^\top]\ . \end{aligned}$$

Putting these expressions together results in (3). ∎

The matrix $L \equiv D - W$ is the *Laplacian* of the weighted graph defined above. Note that $e^\top L = 0$, so L is singular.

In order to define the y_is by minimizing (3), we need to add a constraint to V. From here there are several ways to proceed depending on what is desired.

OLPP We can simply enforce the mapping to be orthogonal, i.e., we can impose the condition $V^\top V = I$. In this case the set V is the eingenbasis associated with the lowest eigenmodes of the matrix

$$C_{lpp} = X(D - W)X^\top \tag{4}$$

We refer to this first option as the method of Orthogonal Locality Preserving Projections. It differs from the original LPP approach, which uses the next option.

LPP We can impose a condition of orthogonality on the projected set: $YY^\top = I$. Note that the rows of Y are orthogonal, which means that the d basis vectors in $\mathcal{R}^n$ on which the x_i's are projected are orthogonal. Alternatively, we can also impose an orthogonality with respect to the weight D: $YDY^\top = I$. (This gives bigger weights to points y_i's for which $d_i \sum_j w_{ij}$ is large).

ONPP to be discussed later uses an option that is close to the first one, but it replaces the matrix $D-W$ with a matrix that is quite different. The first option leads to the standard eigenvalue problem:

$$X(D-W)X^\top v_i = \lambda_i v_i \ . \tag{5}$$

The classical LPP option leads to the generalized eigenvalue problem.

$$X(D-W)Xv_i = \lambda_i XDXv_i. \tag{6}$$

In both cases the smallest d eigenvalues and eigenvectors must be computed.

A slight drawback of the scaling used by classical LPP is that the linear transformation is no longer orthogonal. However, the weights can be redefined (in fact rescaled) a priori so that the diagonal D becomes the identity. We found that the issue of scaling is an important one.

An interesting connection can be made with PCA as was observed in [5]. Using a slightly different argument from [5], suppose we take as W the (dense) matrix $W = \frac{1}{n}ee^\top$. This simply puts the uniform weight $1/n$ to every single pair (i,j) for the full graph. In this case, $D = I$ and the objective function in (4) becomes

$$C_{lpp} = X\left(I - \frac{1}{n}ee^\top\right)X^\top = C_{pca}$$

PCA compute the eigenvectors associated with the largest eigenvalues of a "global" (full) graph. In contrast, methods based on Locality Preservation (such as LPP) compute the eigenvectors associated with the smallest eigenvalues of a "local" (sparse) graph. PCA is likely to be better at conveying global structure, while methods based on preserving the graph will be better at maintaining locality.

3 ONPP

The main idea of ONPP is to seek an orthogonal mapping of a given data set so as to best preserve a graph which describes the local geometry. It is in essence a variation of OLPP discussed earlier, in which the graph is constructed differently.

3.1 The nearest neighbor affinity graph

Consider a dataset represented by the columns of a matrix $X = [x_1, x_2, \ldots, x_n] \in R^{m\times n}$. ONPP begins by building an affinity matrix by computing optimal weights which will relate a given point to its neighbors in some locally optimal way. This phase is identical with that of LLE [7, 11]. The basic assumption is that each data sample along with its k nearest

neighbors (approximately) lies on a locally linear manifold. Hence, each data sample x_i is reconstructed by a linear combination of its k nearest neighbors. The reconstruction errors are measured by minimizing the objective function

$$\mathcal{E}(W) = \sum_i \|x_i - \sum_j w_{ij} x_j\|_2^2. \tag{7}$$

The weight w_{ij} represent the linear coefficient for reconstructing the sample x_i from its neighbors $\{x_j\}$. The following constraints are imposed on the weights:

1. $w_{ij} = 0$, if x_j is not one of the k nearest neighbors of x_i;
2. $\sum_j w_{ij} = 1$, that is x_i is approximated by a convex combination of its neighbors.

Note that the second constraint on the row-sum is similar to rescaling the matrix W in the previous section, so that it yields a D matrix equal to the identity.

In the case when $w_{ii} \equiv 0$, for all i, then the problem is equivalent to that of finding a sparse matrix Z, ($Z \equiv I - W^\top$) with a specified sparsity pattern, which has ones on the diagonal and whose row-sums are all zero. It is interesting to note in passing that very similar problems are encountered when computing preconditioners by sparse approximate inverses, see, e.g., [9].

There is a simple closed-form expression for the weights. Observe at first that this determination of the $w'_{ij}s$ for a given point x_i is a local one, in the sense that it only depends on x_i and its nearest neighbors. Any algorithm for computing the weight will be fairly inexpensive.

Call G the local Grammian matrix associated with point i. The entries of G are defined by

$$g_{pl} = (x_i - x_p)^\top (x_i - x_l) \in R^{k \times k}.$$

Thus, G contains the pairwise inner products among the neighbors of x_i, given that the neighbors are centered with respect to x_i. Denoting by $X^{(i)}$ a system of vectors consisting of x_i and its neighbors, we need to solve the least-squares $(X^{(i)} - x_i e^\top) w_{i,:} = 0$ subject to the constraint $e^\top w_{i,:} = 1$. It can be shown that the solution $w_{i,:}$ of this constrained least squares problem is given by the following formula [7] using the inverse of G,

$$w_{i,:} = \frac{G^{-1} e}{e^\top G^{-1} e} \ . \tag{8}$$

(recall that e is the vector of all ones). The weights w_{ij} satisfy certain optimality properties. They are invariant to rotations, scalings, and translations. As a consequence of these properties the affinity graph preserves the intrinsic geometric characteristics of each neighborhood.

3.2 The algorithm

Assume that each data point $x_i \in R^m$ is mapped to a lower dimensional point $y_i \in R^d$, $d \ll m$. Since LLE seeks to preserve the intrinsic geometric properties of the local neighborhoods, it assumes that the same weights which reconstruct the point x_i by its neighbors in the high

dimensional space, will also reconstruct its image y_i, by its corresponding neighbors, in the low dimensional space. In order to compute the y_i's for $i = 1, \ldots, n$, LLE employs the objective function:

$$\Phi(Y) = \sum_i \|y_i - \sum_j w_{ij} y_j\|_2^2. \tag{9}$$

In this case the weights W are fixed and we need to minimize the above objective function with respect to $Y = [y_1, y_2, \ldots, y_n] \in R^{d \times n}$.

Similar to the case of LPP and OLPP, we need to impose some constraints on the y_is. This optimization problem is formulated under the following constraints in order to make the problem well-posed:

1. $\sum_i y_i = 0$ i.e., the mapped coordinates are centered at the origin and
2. $\frac{1}{n} \sum_i y_i y_i^\top = I$, that is the embedding vectors have unit covariance.

LLE does not impose any specific other constraints on the projected points, it only aims at reproducing the graph. So the objective function (9) is minimized with the above constraints on Y.

Note that $\Phi(Y)$ can be written $\Phi(Y) = \|Y - YW^\top\|_F^2$, so

$$\Phi(Y) = \|Y(I - W^\top)\|_F^2 = tr\left[Y(I - W^\top)(I - W)Y^\top\right] \tag{10}$$

The problem will amount to computing the d eigenvalues of the matrix $M = (I - W^\top)(I - W^\top)^\top$, and the associated eigenvectors.

In ONPP an explicit linear mapping from X to Y is imposed which is in the form (1). So we have $y_i = V^\top x_i,\ i = 1, \ldots, n$ for a certain matrix matrix $V \in R^{m \times d}$ to be determined. In order to determine the matrix V, ONPP imposes the constraint that each data sample y_i in the reduced space is reconstructed from its k neighbors by exactly the same weights as in the input space. This means that we will minimize the same objective function (10) as in the LLE approach, but now Y is restricted to being related to X by (1). When expressed in terms of the unknown matrix V, the objective function becomes

$$\Phi(Y) = \|V^\top X(I - W^\top)\|_F^2 = tr\left[V^\top X(I - W^\top)(I - W)X^\top V\right]\ . \tag{11}$$

If we impose the additional constraint that the columns of V are orthonormal, i.e. $V^\top V = I$, then the solution V to the above optimization problem is the basis of the eigenvectors associated with the d smallest eigenvalues of the matrix

$$\tilde{M} = X(I - W^\top)(I - W)X^\top\ . \tag{12}$$

The assumptions that were made when defining the weights w_{ij} at the beginning of this section, imply that the matrix $I - W$ is singular. In the case when $m > n$ the matrix $\tilde{M}$, which is of size $m \times m$, is at most of rank n and it is therefore singular. In the case when $m \leq n$, $\tilde{M}$ is not necessarily singular. However, we observed in practice that ignoring the smallest eigenvalue of M, which is zero in theory, is helpful. Note that the corresponding eigenvector is not the trivial vector e as is the case in LLE. Note also that the embedding vectors of LLE are obtained by computing the eigenvectors of matrix M associated with its smallest eigenvalues.

Algorithm: **ONPP**
Input: Dataset $X \in R^{m \times n}$ and d: dimension of reduced space.
Output: Embedding vectors $Y \in R^{d \times n}$.
1. Compute the k nearest neighbors of data points.
2. Compute the weights w_{ij} which give the best linear reconstruction of each data point x_i by its neighbors (Equ. (8)).
3. Compute the projected vectors $y_i = V^\top x_i$, where V is determined by computing the $d+1$ eigenvectors of
$$\tilde{M} = X(I - W^\top)(I - W)X^\top$$
associated with smallest eigenvalues.

Table 1: The ONPP algorithm.

An important property of ONPP is that mapping new data points to the lower dimensional space is trivial once the matrix V is determined. Consider a new test data sample x_t that needs to be projected. The test sample is projected onto the subspace $y_t = V^\top x_t$ using the dimensionality reduction matrix V. Therefore, the computation of the new projection simplifies to a matrix vector product.

In terms of Computational cost, the first part of ONPP consists of forming the k-NN graph. This scales as $O(n^2)$. Its second part requires the computation of a few of the smallest eigenvectors of $\tilde{M}$. Observe that in practice this matrix is not computed explicitly. Rather, iterative techniques are used to compute the corresponding smallest singular vectors of matrix $X(I-W)^\top$ [8]. The inner computational kernel of these techniques is the matrix-vector product which scales quadratically with the dimensions of the matrix at hand.

3.3 Discussion

We can also think of developing a technique based on enforcing an orthogonality relationship between the projected points instead of the V's. Making the projection orthogonal will tend to preserve distances and so the overall geometry will be preserved. In contrast, imposing the condition $YY^\top = I$, will lead to a criterion that is similar to that of PCA: the points y_i will tend to be different from one another (because of the orthogonality of the rows of Y). This is precisely what LLE does. In essence, the main difference between LLE and ONPP is in the selection of the orthogonality to enforce.

The two optimization problems are shown below:

$$\begin{array}{lcll} \text{LLE} & : & \min_{Y \in R^{m \times d};\ YY^\top = I} & tr[YMY^\top] \\ \text{ONPP} & : & \min_{Y = V^\top X; V \in R^{m \times d};\ VV^\top = I} & tr[YMY^\top] \end{array} .$$

It is also possible to enforce a linear relation between the Y and X data, but require the same orthogonality as LLE. We will refer to this procedure as *Neighborhood Preserving Projections* (NPP). In NPP, the objective function is the same as with ONPP and is given

by (11). However, the constraint is now $YY^\top = I$ which yields, $V^\top XX^\top V = I$. What this means is that NPP is a linear variant of LLE which makes the same requirement on preserving the affinity graph and obtaining a data set Y which satisfies $YY^\top = I$:

$$\text{NPP} \;:\; \min_{Y=V^\top X; V\in R^{m\times d};\; YY^\top=I} \; tr[YMY^\top]$$

If we define $G = XX^\top$, then this leads to the problem,

$$\min_{V \,\in\, R^{m\times d},\; V^\top GV=I} \; V^\top MV \;. \tag{13}$$

The solution if the above problem can be obtained by solving the generalized eigenvalue problem $(V^\top M)v = \lambda Gv$. We note that in practice, the vectors V obtained in this way need to be scaled, for example, so that their columns have unit 2-norms.

It is interesting to observe that the eigenvectors which will be found are actually generalized singular vectors of the pair $[X(I - W^\top), X]$. In contrast, ONPP requires (standard) left singular vectors of the matrix $X(I - W^\top)$, whereas LLE requires left singular vectors of $I - W^\top$. In the sequel we will focus primarily on ONPP, though the variations to be described can be also be defined for LPP, OLPP, and NPP.

4 Supervised ONPP

ONPP can be implemented in either an unsupervised or a supervised setting. In the later case where the class labels are available, ONPP can be modified appropriately and yield a projection which carries not only geometric information but discriminating information as well. In a supervised setting we first build the data graph $G = (N, E)$, where the nodes N correspond to data samples and an edge $e_{ij} = (x_i, x_j)$ exists if and only if x_i and x_j belong to the same class. In other words, we make adjacent those nodes (data samples) which belong to the same class. Notice that in this case one does not need to set the parameter k, the number of nearest neighbors, and the method becomes fully automatic.

Denote by c the number of classes and n_i the number of data samples which belong to the i-th class. The data graph G consists of c cliques, since the adjacency relationship between two nodes reflects their class relationship. This implies that with an appropriate reordering of the columns and rows, the weight matrix W will have a block diagonal form where the size of the i-th block is equal to the size n_i of the i-th class. In this case W will be of the following form,

$$W = \text{diag}(W_1, W_2, \ldots, W_c).$$

The weights W_i within each class are computed in the usual way, as described by equation (8). The rank of W defined above, is restricted as is explained by the following proposition.

Proposition 4.1 *The rank of $I - W$ is at most $n - c$.*

Proof. Recall that the row sum of the weight matrix W_i is equal to 1, because of the constraint (2). This implies that $W_i e_i = e_i, e_i = [1, \ldots, 1]^\top \in R^{n_i}$. Thus, the following c vectors

$$\begin{bmatrix} e_1 & 0 & \cdots & 0 \\ 0 & e_2 & \cdots & 0 \\ 0 & 0 & \cdots & e_c \end{bmatrix},$$

are linearly independent and belong to the null space of $I - W$. Therefore, the rank of $I - W$ is at most $n - c$. ∎

Consider now the case $m > n$ where the number of samples (n) is less than their dimension (m). This case is known as the *undersampled size* problem. A direct consequence of the above proposition is that in this case, the matrix $\tilde{M} \in R^{m \times m}$ will have rank at most $n - c$. In order to ensure that the resulting matrix $\tilde{M}$ will be nonsingular, we may employ an initial PCA projection that reduces the dimensionality of the data vectors to $n - c$. Call $V_{\rm PCA}$ the dimensionality reduction matrix of PCA. Then the ONPP algorithm is performed and the total dimensionality reduction matrix is given by

$$V = V_{\rm PCA} V_{\rm ONPP},$$

where $V_{\rm ONPP}$ is the dimensionality reduction matrix of ONPP.

5 Kernel ONPP

It is possible to formulate a kernelized version of ONPP. Kernels have been extensively used in the context of Support Vector machines (SVMs) [12]. Essentially, a nonlinear mapping $\Phi : R^m \rightarrow \mathcal{H}$ is employed, where $\mathcal{H}$ is a certain high-dimensional *feature space*. Denote by $\Phi(X) = [\Phi(x_1), \Phi(x_2), \ldots, \Phi(x_n)]$ the transformed dataset in $\mathcal{H}$.

The main idea of Kernel ONPP rests on the premise that the transformation Φ is only known through its Grammian on the data X. In other words, what is known is the matrix K whose entries are

$$K_{ij} \equiv k(x_i, x_j) = \langle \Phi(x_i), \Phi(x_j) \rangle. \tag{14}$$

This is the Gram matrix induced by the kernel $k(x, y)$ associated with the feature space. In fact, another interpretation of the Kernel mapping is that we are defining an alternative inner product in the X-space, which is defined through the inner product of every pair (x_i, x_j) as $< x_i, x_j >= k_{ij}$.

Formally, ONPP can be realized in a kernel form by simply applying it to the set $\Phi(X)$. Observe first that

$$K \equiv \Phi(X)^\top \Phi(X) \ . \tag{15}$$

There are two implications of this definition. The first is that the mapping W has to be defined using this new inner product. The second is that the optimization problem too has to take the inner product into account.

Consider first the graph definition. In the feature space we would like to minimize

$$\sum_{i=1}^{m} \|\Phi(x_i) - \sum_j w_{ij} \Phi(x_j)\|_2^2.$$

This is the same as the cost function (7) evaluated on the set $Z \equiv \Phi(X)$ as desired, and therefore an alternative expression for it is

$$\begin{aligned} \mathcal{E}(W) &= \|\Phi(X)(I - W^\top)\|_F^2 \\ &= tr[(I - W)\Phi(X)^\top \Phi(X)(I - W^\top)] \\ &= tr[(I - W)K(I - W^\top)] \end{aligned}$$

Note that K is dense and $n \times n$. The easiest way to solve the above problem is to extract a low rank approximation to the Grammian K, e.g.,

$$K = US^2U^\top = (US)(US)^\top$$

Where $W \in \mathcal{R}^{m\times k}$ and $S \in \mathcal{R}^{k\times k}$. Then the above problem becomes one of minimizing

$$tr(I-W)USSU^\top(I-W)^\top = \|(I-W)US\|_F^2 = \|SU^\top(I-W^\top)\|_F^2.$$

Therefore, $SU^\top$ replaces X when constructing W.

Consider now the problem of obtaining the projection matrix V in a kernel framework. Observe that if we were to work in feature space, then we would take $Y_\Phi = V^\top\Phi(X)$, where $V \in \mathcal{R}^{L\times d}$, where L is the (typically large and unknown) dimension of the feature space. Now the cost function (11) would become

$$tr\left[V^\top\Phi(X)(I-W^\top)(I-W)\Phi(X)^\top V\right]. \tag{16}$$

Since $\Phi(X)$ is not explicitly known (and is of large dimension) this direct approach does not work.

The first way out is to restrict V to be in the range of $\Phi(X)$. This is natural since each column of V is in $\mathcal{R}^L$ the row-space of $\Phi(X)$. Specifically, we write $V = \Phi(X)Z$ where $Z \in \mathcal{R}^{m\times d}$ and $Z^\top Z = I$. Then (16) becomes

$$tr\left[Z^\top\Phi(X)^\top\Phi(X)(I-W^\top)(I-W)\Phi(X)^\top\Phi(X)Z\right] = tr\left[Z^\top KMKZ\right]\ . \tag{17}$$

In a supervised setting, we need to project a test point x_t onto the space of lower dimension. The dot product of x_t with all low dimensional basis vectors $\Phi(X)Z$ is computed as

$$Z^\top\Phi(X)^\top\Phi(x_t) = Z^\top K(:,x_t)\ . \tag{18}$$

Here matlab notation is used so $K(:,x_t)$ represents the vector $(k(x_j,x_t))_{j=1:n}$.

It is somewhat unnatural to have the matrix K be involved quadratically in the expression (17). Equations (16) suggests that we should really obtain K not K^2, since $\Phi(X)^\top\Phi(X) = K$. For example if $W \equiv 0$, then (16) would become $tr(V^\top KV)$ whereas (17) would yield $tr(Z^\top K^2Z)$. The second solution is to involve an implicit polar decomposition of $\Phi(X)$:

$$\Phi(X) = QS \quad \text{with} \quad S = (\Phi(X)^\top\Phi(X))^{1/2} \quad Q^\top Q = I\ . \tag{19}$$

Note that Q is now an orthogonal basis of the range of $\Phi(X)$, and in this case, when $V = QZ$ then (16) becomes

$$\begin{aligned} &tr\left[Z^\top Q^\top QS(I-W^\top)(I-W)SQ^\top QZ\right] \\ &\quad = tr\left[Z^\top S(I-W^\top)(I-W)SZ\right]. \end{aligned} \tag{20}$$

It is not necessary to compute $S = K^{1/2}$ because the above problem can be solved as a generalized eigenvalue problem instead. Let us set $V_* = SZ$. Then

$$\begin{aligned} &\min_{Z\ \in\mathcal{R}^{m\times d}, Z^\top Z=I} \quad tr\left[Z^\top S(I-W^\top)(I-W)SZ\right] \\ =\ &\min_{V_*\ \in\mathcal{R}^{m\times d}, V_*^\top KV_*=I} \quad tr\left[V_*^\top(I-W^\top)(I-W)V_*\right]\ . \end{aligned}$$

This minimization can be achieved by solving the eigenvalue problem

$$(I - W^\top)(I - W)v = \lambda K v.$$

Now the inner product of x_t with all low dimensional basis vectors QZ becomes (recall that $V_* = SZ$, $Q = \Phi(X)S^{-1}$, and that S is symmetric)

$$\begin{aligned} Z^\top Q^\top \Phi(x_t) &= Z^\top S^{-1}\Phi(X)^\top \Phi(x_t) \\ &= Z^\top S^{-1} S^{-1}\Phi(X)^\top \Phi(x_t) \\ &= V_*^\top K^{-1}\ K(:, x_t)\ . \end{aligned} \tag{21}$$

Matlab notation is used again so $K(:, x_t)$ represents the vector $(k(x_j, x_t))_{j=1:n}$.

Now consider for simplicity the case $d = 1$. Concerning the training points, observe that the projections along the eigenvector v are given by $y = Ka$. Then, notice that minimizing the trace (17) amounts to minimizing $y^\top My$ which leads to the exact same eigenvalue problem solved by LLE. Therefore, LLE (nonlinear) could be viewed as performing ONPP (linear) implicitly in the feature space $\mathcal{H}$.

6 Experimental Results

In this section we evaluate all four linear dimensionality reduction methods LPP, NPP, OLPP and ONPP. We use an implementation of LPP which is publicly available[1]. The implementation of OLPP is based on a slight modification of the publically available LPP code.

6.1 Synthetic data

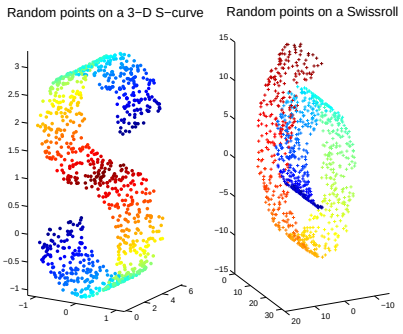

Figure 1: Two examples of data points randomly taken on 3-D manifolds.

Let us first consider two well known synthetic datasets from [11]: the `s-curve`, and the `swissroll`. Figure 1 illustrates the 3-D randomly sampled points on the s-curve and swissroll manifolds. Figures 2 and 3 illustrate the two dimensional projections obtained by the ONPP and LPP methods in the `scurve` and `swissroll` datasets. The affinity graphs

[1] `http://people.cs.uchicago.edu/~xiaofei/LPP.m`

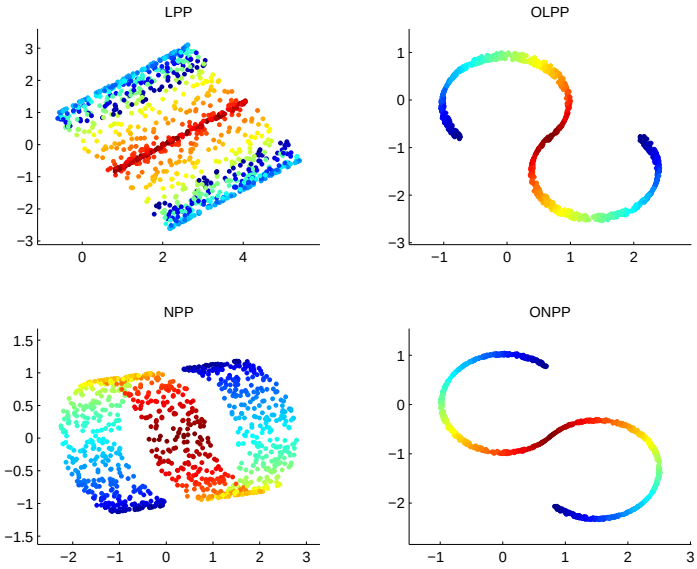

Figure 2: Results of four related methods applied to the S-curve example.

were all constructed using $k = 10$ nearest neighbor points. Observe that the performance of LPP parallels that of NPP and, similarly, the performance of OLPP parallels that of ONPP. Note that all methods preserve locality which is indicated by the color shading. However, the orthogonal methods i.e., OLPP and ONPP preserve global geometric characteristics as well, since they give a faithful projections which convey information about how the manifold is folded in the high dimensional space.

6.2 Digit visualization

The next experiment involves digit visualization. We use 20×16 images of handwritten digits which are publically available from S. Roweis' web page[2]. The dataset contains 39 samples from each class (digits from '0'-'9'). Each digit image sample is represented lexicographically as a high dimensional vector of length 320. For the purpose of comparison with PCA, we first project the dataset in the two dimensional space using PCA and the results are depicted in Figure 4. In the sequel we project the dataset in two dimensions using all four methods. The results are illustrated in Figures 5 (digits '0'-'4') and 6 (digits '5'-'9'). We use $k = 6$ for constructing the affinity graphs of all methods.

Observe that the projections of PCA are spread out since PCA aims at maximizing the variance. However, the classes of different digits seem to heavily overlap. This means that PCA is not well suited for discriminating between data. On the other hand, observe that all the four graph-based methods yield more meaningful projections since samples of the same class are mapped close to each other. This is because these methods aim at preserving locality. Finally, ONPP seems to provide slightly better projections than the other methods since its clusters appear more cohesive.

[2] `http://www.cs.toronto.edu/~roweis/data.html`

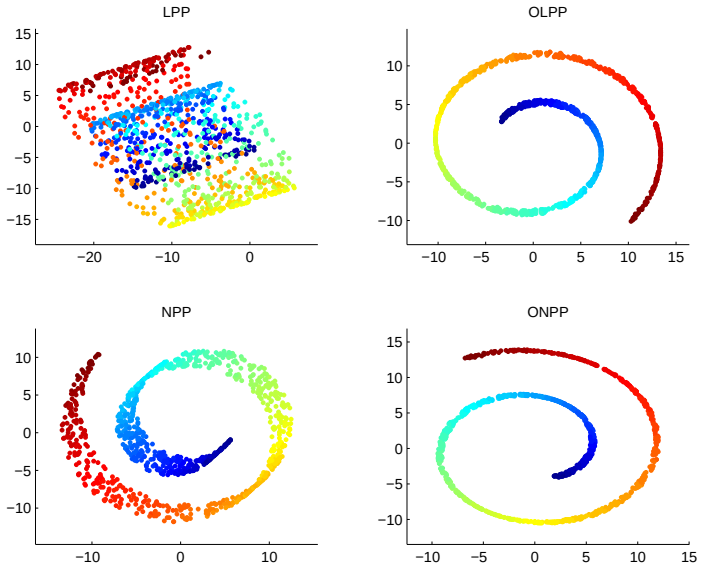

Figure 3: Results of four related methods applied to the Swissroll example.

6.3 Face recognition

In this section we evaluate all methods for the problem of face recognition. We used three datasets which are publically available: UMIST [3], ORL [10] and AR [6]. The size of the images is 112×92 in all datasets. For computational efficiency the images in both databases were downsampled to size 38×31. Thus, each facial image was represented lexicographically as a high dimensional vector of length 1,178. In order to measure the recognition performance, we use a random subset of facial expressions/poses from each subject as training set and the remaining as test set. In order to ensure that our results are not biased from a specific random realization of the training/test set, we perform 20 different random realizations of the training/test sets and we report the average error rate.

We also compare all four methods with Fisherfaces [1], a well known method for face

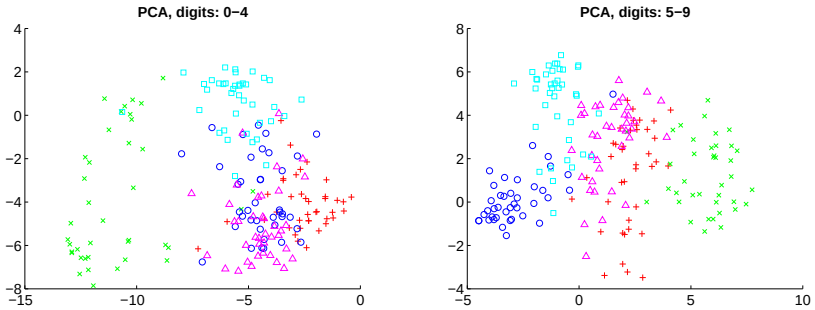

Figure 4: Two dimensional projections of digits using PCA. Left panel: '+' denotes 0, 'x' denotes 1, 'o' denotes 2, '△' denotes 3 and '□' denotes 4. Right panel: '+' denotes 5, 'x' denotes 6, 'o' denotes 7, '△' denotes 8 and '□' denotes 9.

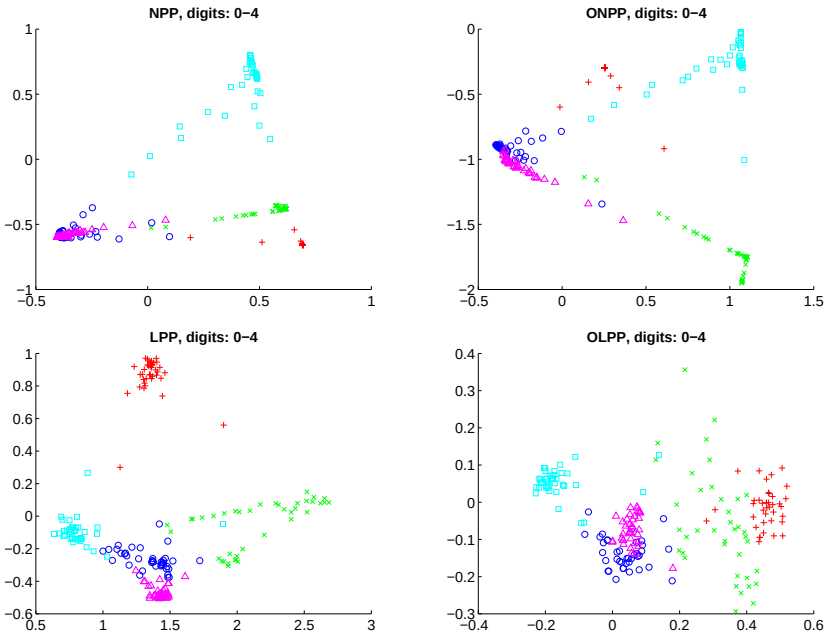

Figure 5: Two dimensional projections of digits using four related methods, where '+' denotes 0, 'x' denotes 1, 'o' denotes 2, '△' denotes 3 and '□' denotes 4.

recognition. Fisherfaces is a supervised method which determines V by using Linear Discriminant Analysis (LDA). LDA works by extracting a set of "optimal" discriminating axes. Assume that we have c classes and that class i has n_i data points. Define the *between-class scatter matrix*

$$S_B = \sum_{i=1}^{c} n_i (\mu^{(i)} - \mu)(\mu^{(i)} - \mu)^\top$$

and the *within-class scatter matrix*

$$S_W = \sum_{i=1}^{c} \left(\sum_{j=1}^{n_i} (x_j^{(i)} - \mu^{(i)})(x_j^{(i)} - \mu^{(i)})^\top \right)$$

where $\mu^{(i)}$ is the centroid of the i-th class. In LDA the columns of V are the eigenvectors associated with largest eigenvalues of the generalized eigenvalue problem $S_B w = \lambda S_W w$. Intuitively, the matrix V of LDA maximizes the ratio of inter-class variance over the intra-class variance.

The tests which follow employ the *supervised versions* of the four methods. This is accomplished by constructing the k-NN graph in a special way which exploits the class labels (see Section 4 for more details on the supervised version of ONPP). In the LPP and OLPP methods, we employ Gaussian weights. We determine the value of the width σ of the Gaussian envelope as follows. First, we sample 1000 points randomly and then compute the pairwise distances among them. Then σ is set equal to half the median of those pairwise distances. This gives a good and reasonable estimate for the value of σ.

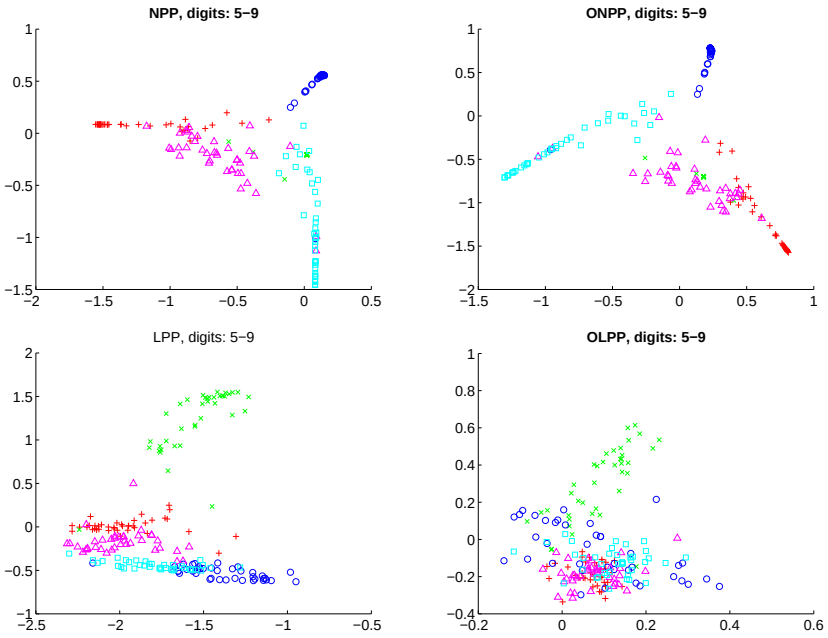

Figure 6: Two dimensional projections of digits using four related methods, '+' denotes 5, 'x' denotes 6, 'o' denotes 7, '△' denotes 8 and '□' denotes 9.

6.4 UMIST

Figure 7: Sample face images from the UMIST database. The number of different poses poses for each subject is varying.

The UMIST database [3] contains 20 people under different poses. The number of different views per subject varies from 19 to 48. We used a cropped version of the UMIST database that is publically available from S. Roweis' web page[3]. Figure 7 illustrates a sample subject from the UMIST database along with its first 20 views. We form the training set by a random subset of 15 different poses per subject (300 images in total) and use the remaining poses as a test set. We experiment with the dimension of the reduced space $d = [10 : 5 : 70]$ (in `MATLAB` notation) and for each value of d we plot the average error rate across 20 random realizations of the training/set set. The results are illustrated in Figure 10.

Concerning the method of Fisherfaces note that there are only $c-1$ generalized eigenvalues, where c is the number of subjects in the dataset. Thus, d cannot exceed $c-1$ and so we plot only the best achieved error rate by Fisherfaces across the various values of d. Observe

[3] `http://www.cs.toronto.edu/~roweis/data.html`

Figure 8: Sample face images from the ORL database. There are 10 available facial expressions and poses for each subject.

Figure 9: Sample face images from the AR database.

again that NPP and LPP have similar performance and that ONPP competes with OLPP and they both outperform the other methods across all values of d. We also report the best error rate achieved by each method and the corresponding dimension d of the reduced space. The results are tabulated in the left portion of Table 2. Notice that PCA works surprisingly well in this database.

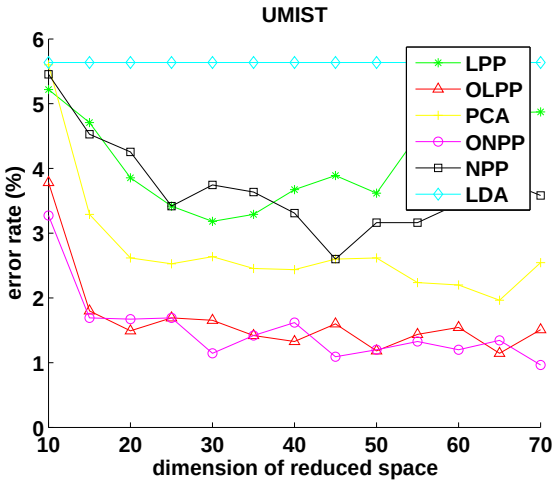

Figure 10: Error rate with respect to the reduced dimension d. Top panel: UMIST database and bottom panel: AR database.

6.5 ORL

The ORL (formerly Olivetti) database [10] contains 40 individuals and 10 different images for each individual including variation in facial expression (smiling/non smiling) and pose. Figure 8 illustrates two sample subjects of the ORL database along with variations in facial expression and pose. We form the training set by a random subset of 5 different facial

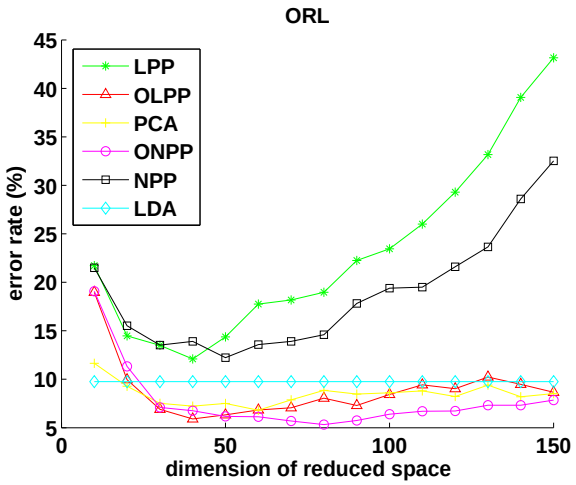

Figure 11: Error rate with respect to the reduced dimension d. Top panel: UMIST database and bottom panel: AR database.

	UMIST		ORL		AR	
	d	error (%)	d	error (%)	d	error (%)
PCA	65	1.96	60	6.8	90	17.72
LDA	30	5.63	70	9.75	100	7.89
LPP	30	3.18	40	12.1	60	8.86
NPP	45	2.6	50	12.22	100	9.33
OLPP	65	1.14	40	5.9	100	4.51
ONPP	70	0.96	80	5.32	100	4.99

Table 2: The best error rate achieved by all methods on the UMIST, ORL and AR databases respectively .

expressions/poses per subject and use the remaining 5 as a test set. We experiment with the dimension of the reduced space $d = [10 : 10 : 150]$ and for each value of d we compute the average error rate across 20 random realizations of the training set.

Figure 11 illustrates the results. Here, LPP and NPP exhibit an unusual behavior: Their error rates initially decrease with the dimension d and then start growing fast after some point. Notice also that the orthogonal methods ONPP and OLPP outperform again the remaining methods and that the former seems to be slightly better than the latter, overall. The best error rates achieved by each method are tabulated in Table 2 along with the corresponding value of d.

6.6 AR

We use a subset of the AR face database [6] which contains 126 subjects under 8 different facial expressions and variable lighting conditions for each individual. Figure 9 depicts two subjects randomly selected from the AR database under various facial expressions and illu-

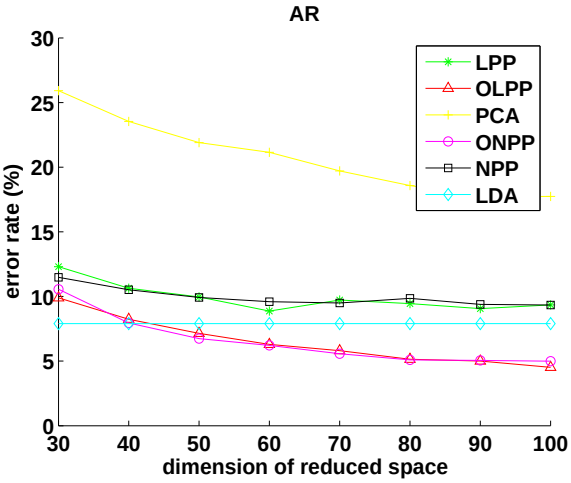

Figure 12: Error rate with respect to the reduced dimension d. Top panel: UMIST database and bottom panel: AR database.

mination. We form the training set by a random subset of 4 different facial expressions/poses per subject and use the remaining 4 as a test set. We plot the error rate across 20 random realizations of the training/test set, for $d = [30 : 10 : 100]$.

The results are illustrated in Figure 12. Once again we observe that ONPP and OLPP outperform the remaining methods across all values of d. In addition, notice that NPP has parallel performance with LPP and they are both inferior to Fisherfaces. Furthermore, Table 2 reports the best achieved error rate and the corresponding value of d. Finally, observe that for this database, PCA does not perform too well. In addition, OLPP and ONPP yield very similar performances for this case.

The above experimental results suggest that the orthogonality of the columns of the dimensionality reduction matrix V is very important for data visualization purposes. This is more evident in the case of face recognition, where this particular feature turned out to be crucial for the performance of the method at hand.

7 Conclusion

The Orthogonal Neighborhood Preserving Projections (ONPP) introduced in this paper is a linear dimensionality reduction technique, which will tend to preserve not only the locality but also the local and global geometry of the high dimensional data samples. It can be extended to a supervised method and it can also be combined with kernel techniques. We introduced three methods with parallel characteristics and compared their performance in both synthetic and real life datasets. We showed that ONPP and OLPP can be very effective for data visualization, and that they can be implemented in a supervised setting to yield a robust recognition technique.

Acknowledgements We are grateful to Prof. D. Boley for his valuable help and insightful discussions on various aspects of the paper.

References

[1] P. Belhumeur, J. Hespanha, and D. Kriegman. Eigenfaces vs. Fisherfaces: Recognition Using Class Specific Linear Projection. *IEEE Trans. Pattern Analysis and Machine Intelligence, Special Issue on Face Recognition*, 19(7):711—20, July 1997.

[2] Y. Bengio, J-F Paiement, P. Vincent, O. Delalleau, N. Le Roux, and M. Ouimet. Out-of-Sample Extensions for LLE, Isomap, MDS, Eigenmaps, and Spectral Clustering. In Sebastian Thrun, Lawrence Saul, and Bernhard Schölkopf, editors, *Advances in Neural Information Processing Systems 16*. MIT Press, Cambridge, MA, 2004.

[3] D. B Graham and N. M Allinson. Characterizing Virtual Eigensignatures for General Purpose Face Recognition. *Face Recognition: From Theory to Applications*, 163:446–456, 1998.

[4] X. He and P. Niyogi. Locality preserving projections. *Advances in Neural Information Processing Systems 16 (NIPS 2003)*, 2003. Vancouver, Canada.

[5] X. He, S. Yan, Y. Hu, P. Niyogi, and H-J Zhang. Face recognition using Laplacianfaces. *IEEE TPAMI*, 27(3):328–340, March 2005.

[6] A.M. Martinez and R. Benavente. The AR Face Database. Technical report, CVC no. 24, 1998.

[7] S. Roweis and L. Saul. Nonlinear Dimensionality Reduction by Locally Linear Embedding. *Science*, 290:2323–2326, 2000.

[8] Y. Saad. *Numerical Methods for Large Eigenvalue Problems*. Halstead Press, New York, 1992.

[9] Y. Saad. *Iterative Methods for Sparse Linear Systems, 2nd edition*. SIAM, Philadelpha, PA, 2003.

[10] F. Samaria and A. Harter. Parameterisation of a Stochastic Model for Human Face Identification. In *2nd IEEE Workshop on Applications of Computer Vision*, Sarasota FL, December 1994.

[11] L. Saul and S. Roweis. Think Globally, Fit Locally: Unsupervised Learning of Nonlinear Manifolds. *Journal of Machine Learning Research*, 4:119–155, 2003.

[12] V. Vapnik. *Statistical Learning Theory*. Wiley, New York, 1998.