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

Effrosyni Kokiopoulou, *Student member, IEEE,* and Yousef Saad.

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.**

Index Terms— **Linear Dimensionality Reduction, Face Recognition, Data Visualization.**

I. Introduction

The problem of dimensionality reduction appears in many fields including data mining, machine learning and computer vision, to name just a few. It is often a necessary preprocessing step in many systems, usually employed for simplification of the data and noise reduction. 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) [1], [2] is a nonlinear dimensionality reduction technique which aims at preserving the local geometries at each neighborhood.

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) [3], 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. Note that the neighborhood sets are not independent. In fact, since there is a great overlap between the neighborhood sets of near-by data samples, it can be deduced that the global geometric characteristics of the data will be preserved as well. This can also be seen from the fact that the mapping is an orthogonal projection. In principle, orthogonal projections, like PCA, will be "blind" to features that are orthogonal to the span of V. However, the projector can be carefully selected in such a way that these features are unimportant for the task at hand. By their linearity, they will also give good representation of the global geometry. One can view this class of methods as a compromise between PCA which emphasizes global structure, and LLE which is based mainly on preserving local structure.

While one is tempted to take examples from the 3-D to 2-D linear projections, this situation provides too simplistic a representation of the complex situations which occur in high dimensional cases. As will be shown experimentally, linear projections can be quite effective for certain tasks such as data visualization. We provide experimental results which support this claim. In particular, experiments will confirm that ONPP can be an effective tool for data visualization purposes and that it may be viewed as a synthesis of PCA and LLE. In addition, ONPP can provide the foundation of nonlinear techniques, such as Kernel methods [4], [5], or Isomap [6]. In particular, we provide a framework which unifies various well-known methods.

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. [7]). In the case of ONPP the projection of a new data sample is straightforward as it simply amounts to a matrix by vector

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

E. Kokiopoulou is with the Swiss Federal Institute of Technology, Lausanne (EPFL), Signal Processing Institute, LTS4 lab, Bat. ELD 241, Station 11; CH 1015 Lausanne; Switzerland. Email: effrosyni.kokiopoulou@epfl.ch.

Y. Saad is with the Department of Computer Science and Engineering; University of Minnesota; Minneapolis, MN 55455. Email: saad@cs.umn.edu.

product.

ONPP shares some properties with Locality Preserving Projections (LPP)[8]. 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. While Gaussian weights can be used in LPP, these are somewhat artificial and require the selection of an appropriate value of the parameter σ, the width of the Gaussian envelope. This issue is often overlooked, though it is crucial for the performance of the method and remains a serious handicap for the use of Gaussian weights. A second significant difference between LPP and ONPP, is that the latter forces the projection to be orthogonal. In LPP, the projection is defined via a certain objective function, whose minimization leads to eigenvectors of a generalized eigenvalue problem.

II. Dimensionality reduction by projection

Given a data set $X = [x_1, x_2, \ldots, x_n] \in R^{m\times n}$, the goal of dimensionality reduction is to produce a set Y which is an accurate representation of X, but which is of dimension d, with $d \ll m$. 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 may include the global geometry, or neighborhood information such as local neighborhoods, distances between data points, or angles formed by adjacent line segments.

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

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

If we denote by $e = [1, \ldots, 1]^\top$ the vector of ones, it can be easily shown that 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.

A. LPP and OLPP

Another related technique is that of *Locality Preserving Projections* (LPP) [8]. LPP projects the data so as to preserve a certain *affinity graph* constructed from the data. The affinity (or adjacency) graph is a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ whose nodes $\mathcal{V}$ are the data samples. The edges of 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. In the latter case it is called the k-NN graph.

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 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. Note that the authors in [9], [8] give a proof for the case of $d = 1$. In what follows, we provide a proof for the general $d > 1$ case.

Theorem 2.1: Let W be a certain symmetric affinity graph, and define $D = \mathrm{diag(d_i)}$ with

$$d_i = \sum_{j=1}^{n} w_{ij}. \tag{3}$$

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

$$E_{lpp} = \mathrm{tr}[Y(D - W)Y^\top] = \mathrm{tr}[V^\top X(D - W)X^\top V] \tag{4}$$

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:

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

Denoting by e_i the i-th canonical vector, we have for the second term,

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

Putting these expressions together results in (4). ■

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_i's by minimizing (4), 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 eigenbasis associated with the lowest eigenmodes of the matrix

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

We refer to this first option as the method of Orthogonal Locality Preserving Projections (OLPP). This option leads to the standard eigenvalue problem:

$$X(D - W)X^\top v_i = \lambda_i v_i \ , \tag{6}$$

and leads to a matrix V with orthonormal columns. The OLPP option is different than 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). The classical LPP option leads to the generalized eigenvalue problem.

$$X(D - W)X^\top v_i = \lambda_i X D X^\top v_i. \tag{7}$$

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 (i.e., the data can be rescaled a priori) so that the diagonal D becomes the identity.

Note that the above OLPP option that is proposed here, is different than the one proposed in [10], which recently came to our attention while this paper was under review. In short, the authors in [10] enforce orthogonality of the v_i's by imposing explicit orthogonality constraints and they propose a solution based on Lagrange multipliers.

An interesting connection can be made with PCA as was observed in [11]. Using a slightly different argument from [11], 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 matrix (5) which defines the objective function becomes

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

PCA computes 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 seeks the largest eigenvalues due to the fact that its goal is to maximize the variance of the projected data. Similarly, LPP seeks the smallest eigenvalues since it targets at minimizing the distance between similar data samples. PCA is likely to be better at conveying global structure, while methods based on preserving the graph will be better at maintaining locality.

III. 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.

A. The nearest neighbor affinity graph

Consider a data set 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 [1], [2].

For completeness, the process of constructing the affinity graph is summarized here; details can be found [1], [2]. 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

$$\mathcal{E}(W) = \sum_i \|x_i - \sum_j w_{ij} x_j\|_2^2. \tag{8}$$

The weights w_{ij} represent the linear coefficients 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 imposing $d_i = 1$, where d_i is defined in eq. (3). Hence, imposing this constraint is equivalent 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.

There is a simple closed-form expression for the weights. It is useful to point out that determining the w_{ij}'s for a given point x_i is a local calculation, in the sense that it only involves x_i and its nearest neighbors. Any algorithm for computing the weights will be fairly inexpensive.

Let G be the local Grammian matrix associated with point i, whose entries 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 [1] which involves the inverse of G,

$$w_{i,:} = \frac{G^{-1} e}{e^\top G^{-1} e} \ . \tag{9}$$

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

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

$$\mathcal{F}(Y) = \sum_i \|y_i - \sum_j w_{ij} y_j\|_2^2. \tag{10}$$

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, some constraints must be imposed on the y_i's. 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 other specific constraints on the projected points, it only aims at reproducing the graph. So the objective function (10) is minimized with the above constraints on Y.

```
Algorithm: ONPP
Input: Data set $X \in R^{m\times n}$ 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. (9)).
3. Compute $V$ the matrix whose column vectors are the $d$
   eigenvectors of
          $\tilde{M} = X(I - W^\top)(I - W)X^\top$
   associated with 2nd to $(d+1)$st smallest eigenvalues.
4. Compute the projected vectors $y_i = V^\top x_i$.
```

TABLE I
THE ONPP ALGORITHM.

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

$$\begin{aligned} \mathcal{F}(Y) &= \|Y(I - W^\top)\|_F^2 \\ &= \operatorname{tr}\left[Y(I - W^\top)(I - W)Y^\top\right]. \end{aligned} \tag{11}$$

The problem will amount to computing the d smallest 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 (11) 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

$$\begin{aligned} \mathcal{F}(Y) &= \|V^\top X(I - W^\top)\|_F^2 \\ &= \operatorname{tr}\left[V^\top X(I - W^\top)(I - W)X^\top V\right]. \end{aligned} \tag{12}$$

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

$$\tilde{M} = X(I - W^\top)(I - W)X^\top = XMX^\top \ . \tag{13}$$

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 $m \le n$, $\tilde{M}$ is not necessarily singular. However, it can be observed in practice that ignoring the smallest eigenvalue of $\tilde{M}$, is helpful. This is explained in detail in Section III-C. Note that the embedding vectors of LLE are obtained by computing the eigenvectors of M associated with its smallest eigenvalues. This is to be contrasted with ONPP which computes these vectors as $V^\top X$, where V is the set of eigenvectors of $\tilde{M}$ associated with its smallest eigenvalues.

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 using the dimensionality reduction matrix V, so

$$y_t = V^\top x_t. \qquad (14)$$

Therefore, mapping the new data point reduces to a simple 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 $X(I-W)^\top$ [12]. The main computational operation of these techniques is the matrix-vector product which scales quadratically with the dimensions of the matrix at hand.

C. 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 for data points x_i, x_j whose difference $x_i - x_j$ is close to the subspace $span(V)$. Because of linearity, the overall geometry will also tend to 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 maximum variance criterion is also used by LLE. 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:

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

Yet, another point of view is to think in terms of null spaces or approximate null spaces of the matrix $I - W^\top$. LLE builds a matrix W so that X is approximately a left null space for $I - W^\top$, i.e., so that $X(I - W^\top)$ is close to zero. Then, in a second step, it tries to find a $d \times n$ matrix Y so that Y is an approximate null space for $I - W^\top$, by minimizing $\|Y(I-W^\top)\|_F^2 = \text{tr}(YMY^\top)$. The second step of ONPP tries also to find Y so that it is close to a null space for $I - W^\top$, but it does so by restricting the reduced dimension data to be an orthogonal projection of the original data. Interestingly, when $X(I - W^\top)$ is small then so is $V^\top X(I - W^\top)$. If the rows of X happen to be linearly dependent (or very close to being linearly dependent), then a zero row (or a very close to zero row) will appear in the projected data Y. This situation indicates redundancies in the information given on the data. A result is that a linear combination of this information (rows of X) will be zero and this means that a zero row will result in the projected data Y. This zero row should be ignored. This is the reason why one should always discard eigenvectors associated with very small eigenvalues.

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 (12). 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 $YY^\top = I$:

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

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

$$\min_{V \in R^{m\times d},\ V^\top GV = I} \quad \text{tr}[V^\top \tilde{M} V] \ . \qquad (15)$$

The solution of the above problem can be obtained by solving the generalized eigenvalue problem $\tilde{M}v = \lambda Gv$. 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.

IV. 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 $\mathcal{G} = (\mathcal{N}, \mathcal{E})$, where the nodes $\mathcal{N}$ 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 (9). 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_{PCA} the dimensionality reduction matrix of PCA. Then the ONPP algorithm is performed and the total dimensionality reduction matrix is given by

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

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

V. 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), see, e.g., [4], [5]. 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 data set 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

$$K_{ij} \equiv k(x_i, x_j) = \langle \Phi(x_i), \Phi(x_j) \rangle. \tag{16}$$

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 expressed 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 $\Phi(X)$. Define

$$K \equiv \Phi(X)^\top \Phi(X) \ . \tag{17}$$

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.

A. Computation of the graph weights

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 (8) evaluated on the set $\Phi(X)$ as desired, and therefore an alternative expression for it is

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

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 $U \in \mathcal{R}^{n \times \ell}$ and $S \in \mathcal{R}^{\ell \times \ell}$. Then the above problem becomes one of minimizing

$$\begin{aligned} \mathcal{E}(W) &= \text{tr}(I - W)USSU^\top (I - W)^\top & (18) \\ &= \|(I - W)US\|_F^2 & (19) \\ &= \|SU^\top (I - W^\top)\|_F^2. & (20) \end{aligned}$$

Therefore, W is constructed similarly as was described in Section III-A, but now $SU^\top$ replaces X.

Note that the low rank approximation of K is suggested above mostly for computational efficiency. One may well choose to use $\ell = n$ and in this case the resulting graph weights will be exact.

B. Computation of the projection matrix

Consider now the problem of obtaining the projection matrix V in a kernel framework. Formally, if we were to work in feature space, then the projection would take the form $Y = V^\top \Phi(X)$, with $V \in \mathcal{R}^{L \times d}$, where L is the (typically large and unknown) dimension of the feature space. Now the cost function (12) would become

$$\mathcal{F}(Y) = \text{tr}\left[V^\top \Phi(X) M \Phi(X)^\top V\right], \tag{21}$$

where we have used that $M = (I - W^\top)(I - W)$. Since $\Phi(X)$ is not explicitly known (and is of large dimension) this direct approach does not work. We propose two different approaches to attack this problem.

a) Strategy 1: 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 $Z \in \mathcal{R}^{n \times d}$ is to be determined and $Z^\top Z = I$. Then (21) becomes

$$\begin{aligned} \mathcal{F}(Y) &= \text{tr}\left[Z^\top \Phi(X)^\top \Phi(X) M \Phi(X)^\top \Phi(X) Z\right] \\ &= \text{tr}\left[Z^\top KMKZ\right] \ . & (22) \end{aligned}$$

Thus, Z is determined by the eigenvectors of KMK corresponding to its smallest eigenvalues.

In a testing phase, we need to project a test point x_t onto the space of lower dimension, i.e., we need to generalize (14). This is done by noting that the projection is performed from the feature space, so we now need to project $\Phi(x_t)$ using the matrix V:

$$y_t = V^\top \Phi(x_t) = Z^\top \Phi(X)^\top \Phi(x_t) = Z^\top K(\cdot, x_t) \ . \tag{23}$$

Here the notation $K(\cdot, x_t)$ represents the vector $(k(x_j, x_t))_{j=1:n}$.

b) Strategy 2: It is somewhat unnatural that the matrix K is involved quadratically in the expression (22). Equation (21) suggests that we should really obtain K not K^2, since $\Phi(X)^\top \Phi(X) = K$. For example, in the trivial case when $W \equiv 0$, then (21) would become $\text{tr}(V^\top \Phi(X) \Phi(X)^\top V)$ whereas (22) would yield $\text{tr}(Z^\top K^2 Z)$. The second solution is to exploit an implicit QR factorization (or an implicit polar decomposition) of $\Phi(X)$. In the following we will employ a QR factorization of the form:

$$\Phi(X) = QR \tag{24}$$

where R is upper triangular and Q is unitary i.e., $Q^\top Q = I$. This factorization is only implicit since $\Phi(X)$ is not available. Note that

$$R^\top R = \Phi(X)^\top \Phi(X) = K \tag{25}$$

so that $R^\top R$ is the Cholesky factorization of K. In addition, Q is now an orthogonal basis of the range of $\Phi(X)$, so that we can use as a projector in feature space a matrix of the form $V = QZ$, with $Z \in \mathcal{R}^{n\times d}$, $Z^\top Z = I$. The projected data in reduced space is

$$Y = V^\top \Phi(X) = Z^\top Q^\top \Phi(X) = Z^\top Q^\top QR = Z^\top R. \tag{26}$$

In this case, the objective function (21) becomes

$$\begin{aligned}\mathcal{F}(Y) &= \text{tr}\left[Z^\top R(I-W^\top)(I-W)R^\top Z\right] \\ &= \text{tr}\left[Z^\top R\ M\ R^\top Z\right]. \end{aligned}\tag{27}$$

As a result the columns z of the optimal Z are just the set of eigenvectors of the problem

$$\left[R(I-W^\top)(I-W)R^\top\right] z = \lambda z \tag{28}$$

associated with the smallest d eigenvalues. The matrix R can be obtained in practice from the Cholesky factorization of K. However, as we show in the sequel, the problem can also be reformulated to avoid the explicit computation of R.

Indeed, let z be a column of Z, an eigenvector of the matrix $R(I-W^\top)(I-W)R^\top$ associated with some eigenvalue λ. Define $y = R^\top z$ and observe that y is a *transposed row* (a row written as a column vector) of the reduced dimension matrix, $Y = Z^\top R$, per equation (26). We then have:

$$\begin{aligned} R(I-W^\top)(I-W)R^\top z &= \lambda z \quad \rightarrow \\ R^\top R(I-W^\top)(I-W)R^\top z &= \lambda R^\top z \quad \rightarrow \\ K\left[(I-W^\top)(I-W)\right] y &= \lambda y. \end{aligned}\tag{29}$$

Thus, the eigenvectors of $K\left[(I-W^\top)(I-W)\right]$ associated with the smallest d eigenvalues will directly yield the *transposed rows* of the sought projected data Y. In other words, the rows of Y can be directly computed at the smallest *left* eigenvectors of the matrix $(I-W^\top)(I-W)K$. Though the matrix in (29) is nonsymmetric, the problem is similar to the eigenvalue problem $My = \lambda K^{-1} y$ and therefore, the eigenvectors are orthogonal with respect to the K^{-1}-inner product. Using this observation, one may compute directly the projected data set Y, without computing explicitly the matrix R.

Now consider again the testing phase and the analogue of (14). Noting that $Q = \Phi(X)R^{-1}$ we write

$$\begin{aligned} y_t &= V^\top \Phi(x_t) = Z^\top Q^\top \Phi(x_t) \\ &= Z^\top R^{-\top}\Phi(X)^\top \Phi(x_t) \\ &= Z^\top R^{-\top}\ K(\cdot, x_t) \qquad (30)\\ &= Z^\top R(R^{-1}R^{-\top})\Phi(X)^\top\Phi(x_t) \\ &= YK^{-1}\ K(\cdot, x_t)\ . \qquad (31) \end{aligned}$$

Equations (30) and (31) provide two alternative ways of computing y_t, one for when Z is computed by (28) and the other for when Y is computed directly by (29). In either case, the computation will be cubic in n, so this approach is bound to be limited to relatively small data sets. If d is very small and the size of the test data is large, it is of course more economical to compute $Z^\top R^{-\top} = (R^{-1}Z)^\top$ in (30) once and for all at the outset. Similarly, for (31), $YK^{-1} = (K^{-1}Y^\top)^\top$ can also be computed once for all training data. Note that in practice, we don't compute the inverse explicitly, but solve d linear systems instead (one for each different right hand side). Thus, in both cases, d linear systems need to be solved (since both Z and $Y^\top$ have d columns).

C. Discussion

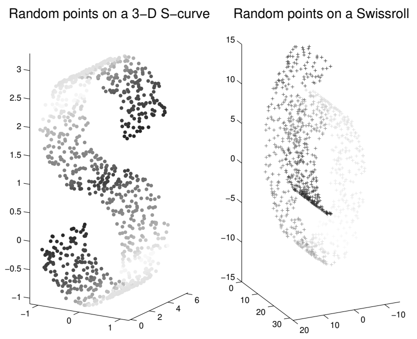


Fig. 1. Two examples of data points randomly taken on 3-D manifolds.

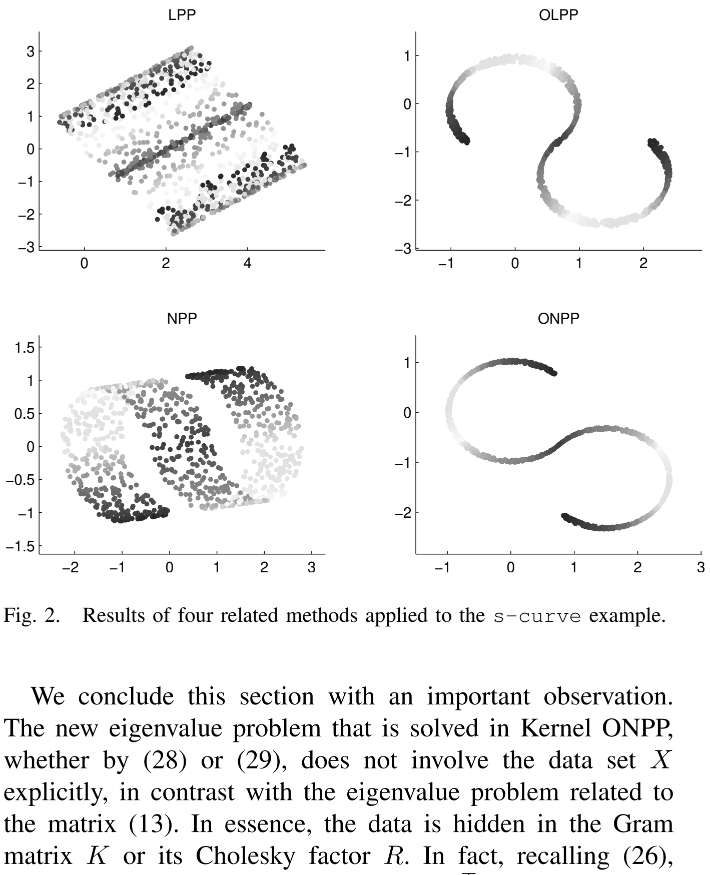


Fig. 2. Results of four related methods applied to the `s-curve` example.

We conclude this section with an important observation. The new eigenvalue problem that is solved in Kernel ONPP, whether by (28) or (29), does not involve the data set X explicitly, in contrast with the eigenvalue problem related to the matrix (13). In essence, the data is hidden in the Gram matrix K or its Cholesky factor R. In fact, recalling (26), we observe that (27) is simply $\text{tr}(YMY^\top)$ and minimizing this trace subject to the condition $Z^\top Z = I$ is equivalent to solving

$$\min_{\substack{Y\in\ \mathcal{R}^{m\times d} \\ YK^{-1}Y^\top = I}} \text{tr}\left[YMY^\top\right]. \tag{32}$$

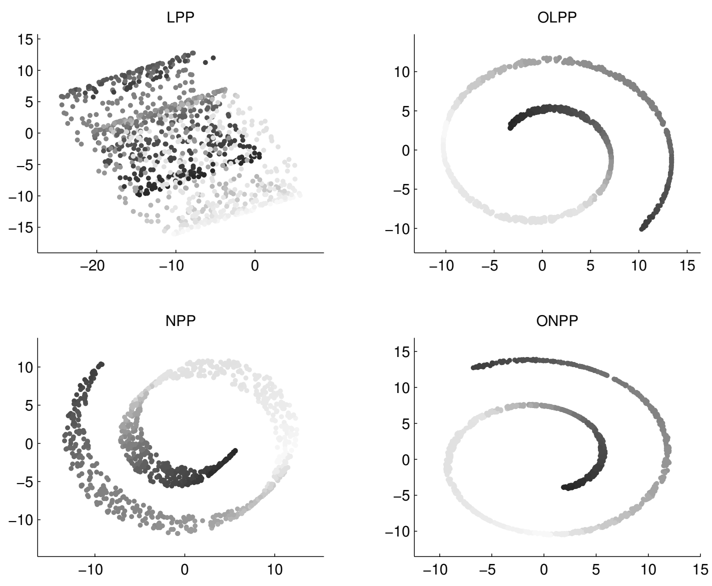


Fig. 3. Results of four related methods applied to the `swissroll` example.

Recalling also the LLE problem from Section III-C, this shows that in effect, *Kernel ONPP is mathematically equivalent to LLE with the K^{-1}-inner product.* For example, LLE can be obtained by defining $K = I$ as a particular case of Kernel ONPP.

This can be pursued a little further by considering the objective function (27) which involves a factor R such that $R^\top R = \Phi(X)^\top \Phi(X)$. Of course we can define R by using other factorizations (for instance, we mentioned above the polar decomposition). Hence, if we use for K the Grammian $X^\top X$, then we might define R to be simply X, since $R^\top R = K$, even though R is no longer an $n \times n$ matrix. The dimension of Z must be changed accordingly to being $m \times d$. This will yield the standard ONPP according to (27). This unconventional extension, allowing R to be an $p \times n$ matrix and Z a $p \times d$ matrix can open some interesting connections. In particular it puts under the same framework LLE, Laplacian eigenmaps [9] alongside with OLPP, and ONPP, by simply changing the matrices R and M. Under this extension, the distinction between LLE, Laplacian Eigenmaps, ONPP, OLPP, and Kernel ONPP, lies in the definitions of the matrices R, and M. Table II indicates the connections between the different methods and the corresponding choices of matrices R and M.

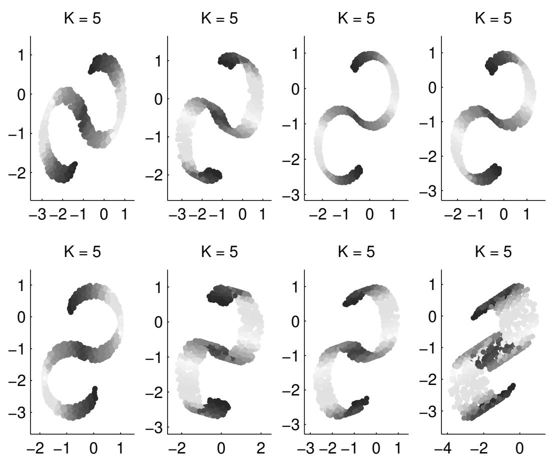
VI. Experimental Results

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.

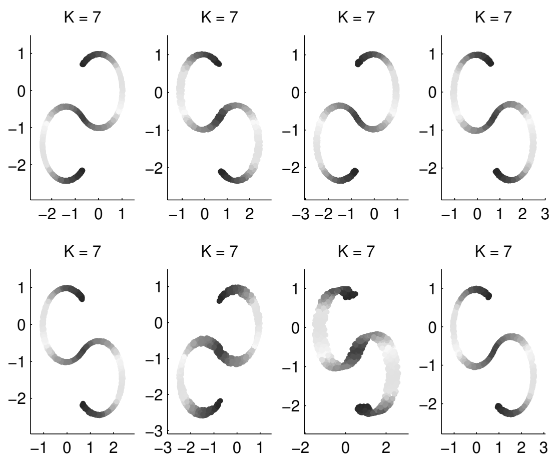
A. Synthetic data

Let us first consider two well known synthetic data sets from [2]: the `s-curve`, and the `swissroll`. Figure 1 illustrates the 3-D randomly sampled points ($n = 1000$) on the `s-curve` and `swissroll` manifolds. Figures 2 and 3 illustrate the two dimensional projections obtained by all

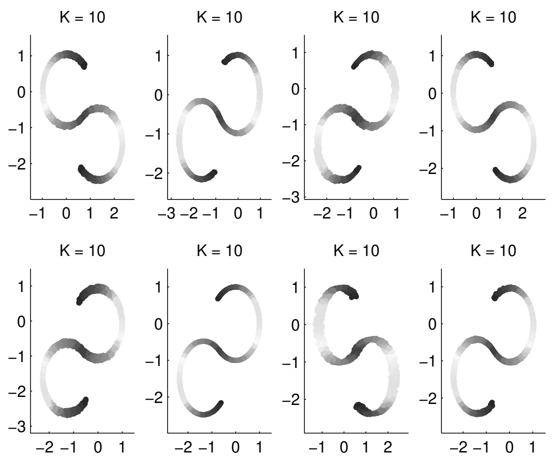
[1]`http://people.cs.uchicago.edu/∼xiaofei/LPP.m`



(a) $k = 5$

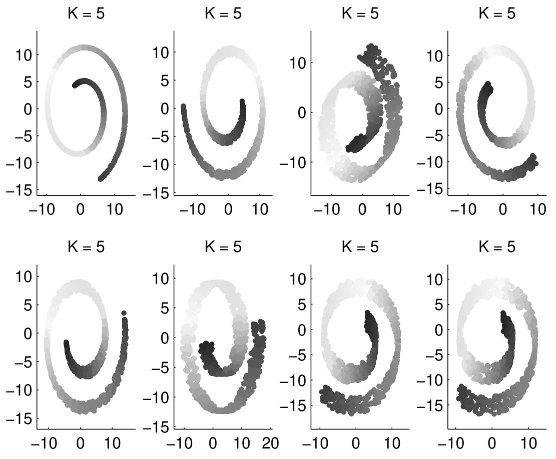


(b) $k = 7$

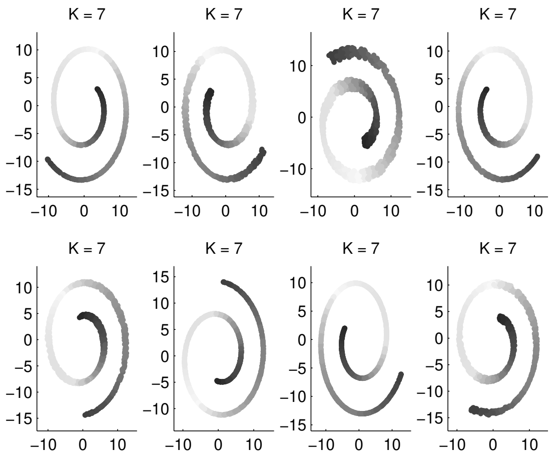


(c) $k = 10$

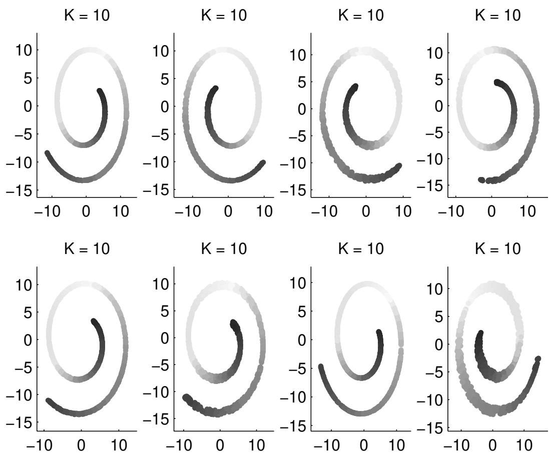
Fig. 4. Behavior of ONPP under different values of k on the `s-curve` data set.



(a) $k = 5$



(b) $k = 7$



(c) $k = 10$

Fig. 5. Behavior of ONPP under different values of k on the `swissroll` data set.

methods in the `s-curve` and `swissroll` data sets. The affinity graphs 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 gray scale darkness value. However, the orthogonal methods i.e., OLPP and ONPP preserve global geometric characteristics as well, since they give faithful projections which convey information about how the manifold is folded in the high dimensional space. This may be the result of the great overlap among the neighbor sets of data samples that are close by.

Method	Matrix M	Matrix R
LLE	$(I - W_a^\top)(I - W_a)$	I
Laplacian Eigenmaps	$(D - W_L^\top)$	I
ONPP	$(I - W_a^\top)(I - W_a)$	X
OLPP	$(D - W_L^\top)$	X
K-ONPP with Kernel K	$(I - W^\top)(I - W)$	$R^\top R = K$ (Chol.)

TABLE II

DIFFERENT METHODS AND THE CORRESPONDING CHOICES FOR THE MATRICES R AND M. THE MATRIX W_a CORRESPONDS TO THE AFFINITY GRAPH IN LLE AND ONPP, SEE SECTION III-A FOR DETAILS. THE MATRIX $D - W_L$ IS THE LAPLACIAN GRAPH USED IN LAPLACIAN EIGENMAPS AND OLPP, SEE SEC. II-A AND [9], [8] FOR DETAILS.

Figures 4 and 5 illustrate the sensitivity of ONPP with respect to random realizations of the data set, for different values of k, for the `s-curve` and the `swissroll` manifolds respectively. We test with a few representative values of k and we compute eight projections for eight different random realizations of the data sets. The number of samples was set to $n = 1000$. Notice that when k is small, the k-NN graph is not able to capture effectively the geometry of the data set. In some cases this results in the method yielding slightly different projections for different realizations of the data set. However, as k increases, the k-NN graph captures more effectively the data geometry and ONPP yields a stable result across the different realizations of the data set.

B. *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 data set 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 data set in the two dimensional space using PCA and the results are depicted in Figure 6. In the sequel we project the data set in two dimensions using all four methods. The results are illustrated in Figures 7 (digits '0'-'4') and 8 (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

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

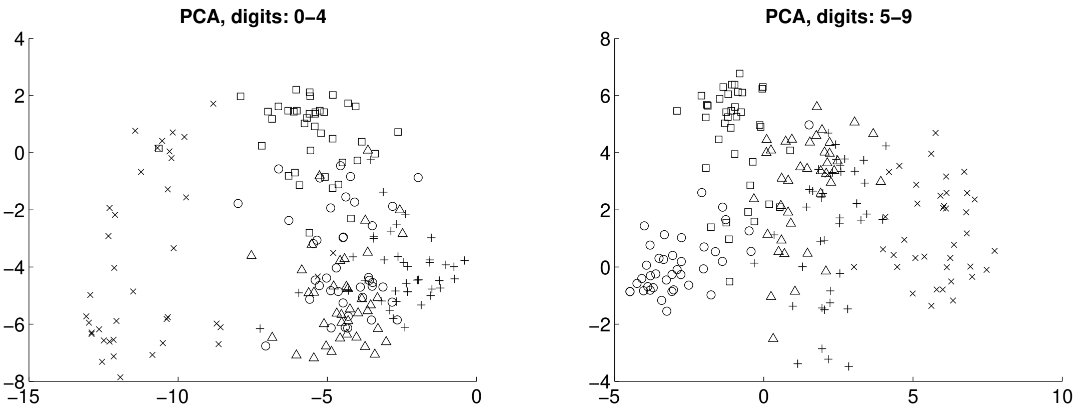


Fig. 6. Two dimensional projections of digits using PCA. Left panel: ‘+’ denotes 0, ‘x’ denotes 1, ‘o’ denotes 2, ‘△’ denotes 3 and ‘□’ denotes 4. Right panel: ‘+’ denotes 5, ‘x’ denotes 6, ‘o’ denotes 7, ‘△’ denotes 8 and ‘□’ denotes 9.

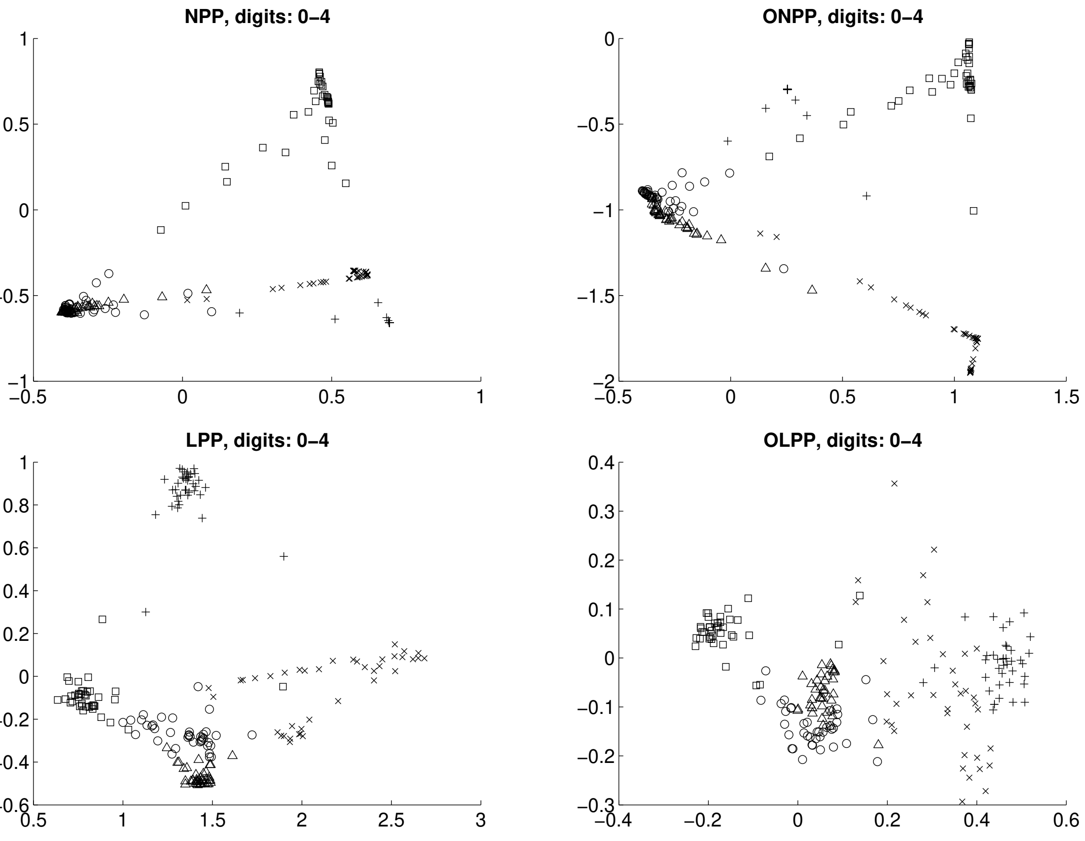


Fig. 7. Two dimensional projections of digits using four related methods, where ‘+’ denotes 0, ‘x’ denotes 1, ‘o’ denotes 2, ‘△’ denotes 3 and ‘□’ denotes 4.

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.

C. Face recognition

In this section we evaluate all methods for the problem of face recognition. We used three data sets which are publically available: UMIST [13], ORL [14] and AR [15]. The size of the images is 112×92 in all data sets. As is common practice the images in all databases were downsampled to size 38×31, for computational efficiency. 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. The test samples are projected in the reduced space using the

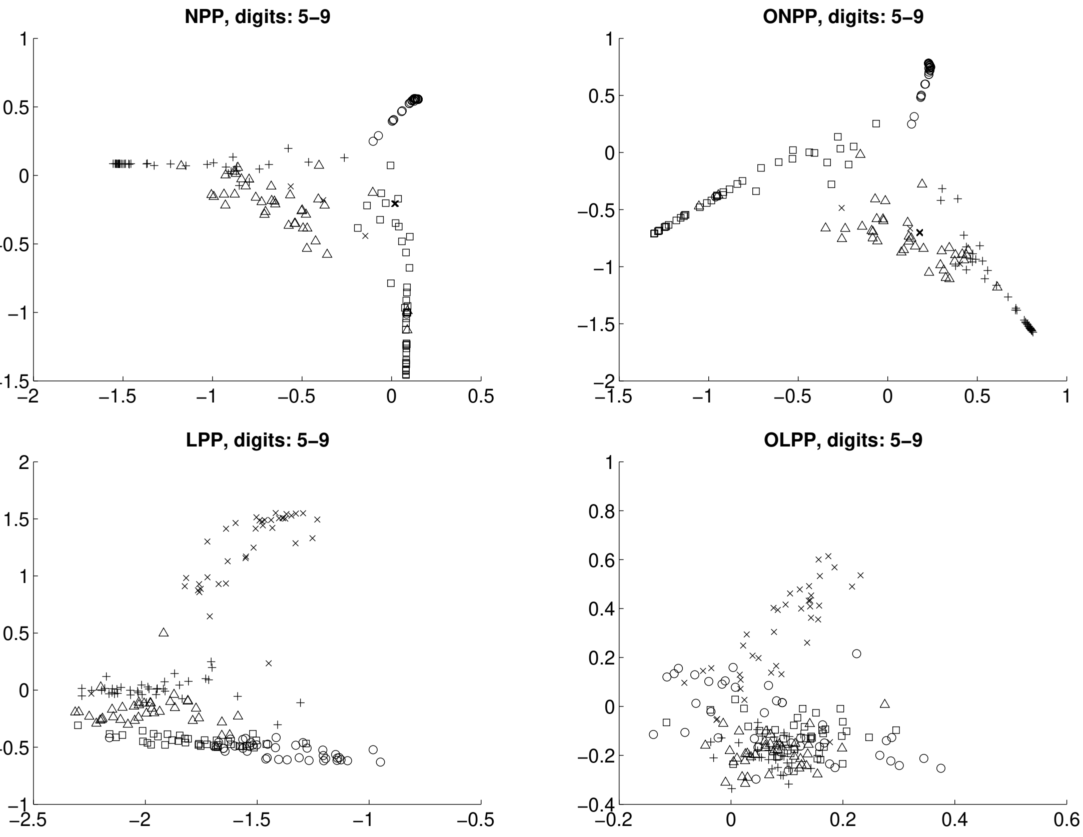


Fig. 8. Two dimensional projections of digits using four related methods, '+' denotes 5, 'x' denotes 6, 'o' denotes 7, '△' denotes 8 and '□' denotes 9.

Fig. 9. Sample face images from the UMIST database. The number of different poses poses for each subject is varying.

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

Fig. 11. Sample face images from the AR database. Facial expressions from left to right: 'natural expression', 'smile', 'anger', 'scream', 'left light on', 'right light on', 'all side lights on' and 'wearing sun glasses'.

dimensionality reduction matrix V which is learned from the training samples. Then, recognition is performed in the low dimensional space using nearest-neighbor (NN) classification. 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 [16], a well known method for face 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 and μ the global centroid. In LDA the columns of V are the eigenvectors associated with largest eigenvalues of the generalized eigenvalue problem

$$S_B w = \lambda S_W w. \tag{33}$$

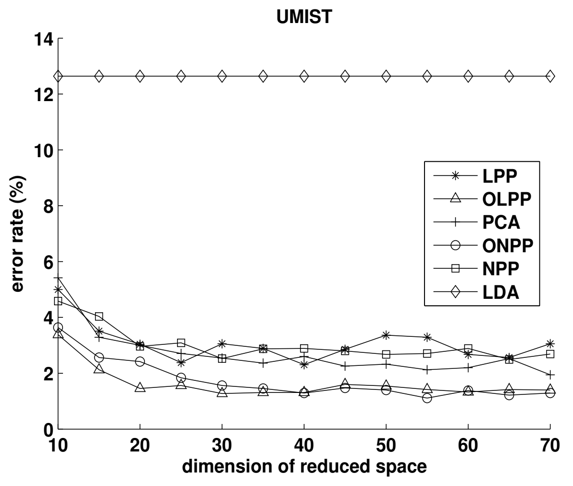


Fig. 12. Error rates with respect to the reduced dimension d, for the UMIST data set.

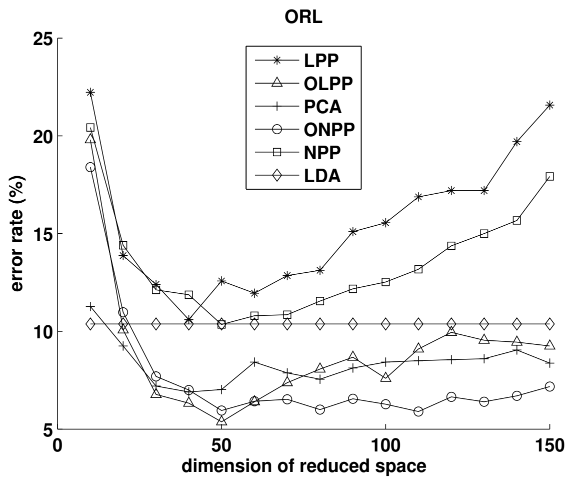


Fig. 13. Error rates with respect to the reduced dimension d, for the ORL data set.

Intuitively, the matrix V of LDA maximizes the ratio of inter-class variance over the intra-class variance. Note that the rank of S_B is at most $c-1$, which implies that the above problem has only $c-1$ generalized eigenvalues. Therefore, LDA can yield at most $c-1$ discriminant axes.

We have observed experimentally that employing *supervised graphs* boosts the classification performance of the methods. This is also a common practice (e.g., see also [10]). Recall that in this case the affinity graph is constructed in a special way which exploits the class labels (see Section IV for more details on the supervised version of ONPP). For example, the best performance of the supervised ONPP on the UMIST dabase (see description later) is 1.11% and is reached at $d = 55$. However the best performance of the unsupervised ONPP is 5.6% and is reached at $k = 20$, for the same d. Thus, we have chosen to use the supervised versions of the four methods in the tests which follow. Note that with the supervised construction of the affinity graph, the parameter k need not be determined by the user, since it is set automatically to be the cardinality of each class.

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.

1) UMIST: The UMIST database [13] 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 9 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 from 10 to 70 with step 5. 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 12.

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

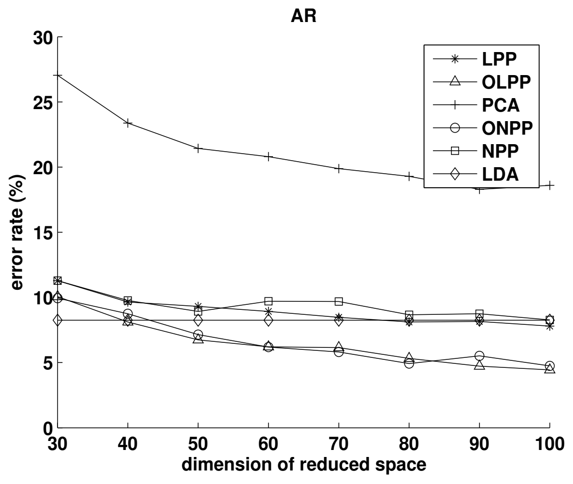


Fig. 14. Error rate with respect to the reduced dimension d, for the AR data set.

Concerning the method of Fisherfaces note that there are only $c-1$ generalized eigenvalues, where c is the number of subjects in the data set. 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 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 III. Notice that PCA works surprisingly well in this database.

2) ORL: The ORL (formerly Olivetti) database [14] contains 40 individuals and 10 different images for each individual including variation in facial expression (smiling/non smiling) and pose. Figure 10 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

	UMIST		ORL		AR	
	d	error (%)	d	error (%)	d	error (%)
PCA	70	1.94	40	6.9	90	18.29
LDA	20	12.63	70	10.37	90	8.25
LPP	40	2.31	40	10.6	100	7.79
NPP	65	2.49	50	10.35	100	8.27
OLPP	30	1.27	50	5.38	100	4.44
ONPP	55	1.11	110	5.9	100	4.74

TABLE III
THE BEST ERROR RATE ACHIEVED BY ALL METHODS ON THE UMIST, ORL, AND AR DATABASES RESPECTIVELY.

facial expressions/poses per subject and use the remaining 5 as a test set. We experiment with the dimension of the reduced space d from 10 to 150 with step 10. For each value of d we compute the average error rate across 20 random realizations of the training set.

Figure 13 illustrates the results. Here, LPP and NPP exhibit an unusual behavior: Their error rates initially decrease with the dimension d and then start growing 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 III along with the corresponding value of d.

3) AR: We use a subset of the AR face database [15] which contains 126 subjects under 8 different facial expressions and variable lighting conditions for each individual. Figure 11 depicts two subjects randomly selected from the AR database under various facial expressions and illumination. 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 different values of d between 30 and 100 with step 10.

The results are illustrated in Figure 14. 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 competitive to Fisherfaces. Furthermore, Table III reports the best achieved error rate and the corresponding value of d. Finally, observe that for this database, PCA has poor performance. In addition, OLPP and ONPP yield very similar performances for this case.

In all previous experiments, we observe a consistent superiority in the performance of the orthogonal methods i.e., ONPP and OLPP versus their non-orthogonal counterparts i.e., NPP and LPP. Thus, the experimental results suggest that the orthogonality of the columns of the dimensionality reduction matrix V is important for data visualization and classification 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.

It is interesting to observe that that "neighborhood-based methods", i.e., LPP, OLPP, ONPP, work rather well in spite of some intrinsic geometric limitations. Specifically, it is difficult to capture the local (as well as global) geometry of a complex data set in high dimensional spaces. For example, a cusp on a sharp curvature on a high-dimensional manifold would need a high value of k to be well-captured. The choice of k, or more generally, the means in which geometry can be better represented in such situations deserves further study. Note for example, that PCA did quite well on at least two examples, suggesting that for these test cases, the local geometry becomes harder to capture by neighborhood-based methods and easier to capture by PCA.

It appears from the experiments shown here that the weighted graph used by ONPP (based on the one in LLE) to represent locality does better that the simpler technique used by OLPP (based on the one in LPP). However, this comparison does not take cost into account. When cost is taken into account, the comparison may not be so clear since a larger k can be taken for LPP/OLPP to compensate for the higher cost of the mapping of LLE/ONPP.

VII. 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 data sets. 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] S. Roweis and L. Saul. Nonlinear Dimensionality Reduction by Locally Linear Embedding. *Science*, 290:2323–2326, 2000.
[2] L. Saul and S. Roweis. Think Globally, Fit Locally: Unsupervised Learning of Nonlinear Manifolds. *Journal of Machine Learning Research*, 4:119–155, 2003.
[3] E. Kokiopoulou and Y. Saad. Orthogonal Neighborhood Preserving Projections. *IEEE Int. Conf. on Data Mining (ICDM)*, November 2005.
[4] K. R. Müller, S. Mika, G. Ratsch, K. Tsuda, and B. Schölkopf. An Introduction to Kernel-based Learning Algorithms. *IEEE Transactions on Neural Networks*, 12:181–201, 2001.
[5] V. Vapnik. *Statistical Learning Theory*. Wiley, New York, 1998.
[6] V. de Silva J. B. Tenenbaum and J. C. Langford. A Global Geometric Framework for Nonlinear Dimensionality Reduction. *Science*, 290(5500):2319–2323, 2000.
[7] 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.
[8] X. He and P. Niyogi. Locality Preserving Projections. *Advances in Neural Information Processing Systems 16 (NIPS 2003)*, 2003. Vancouver, Canada.
[9] M. Belkin and P. Niyogi. Laplacian Eigenmaps for Dimensionality Reduction and Data Representation. *Neural Comput.*, 15(6):1373–1396, 2003.
[10] D. Cai and X. He. Orthogonal Locality Preserving Indexing. ACM SIGIR, Salvador, Brazil, August 15-19, 2005.
[11] X. He, S. Yan, Y. Hu, P. Niyogi, and H-J Zhang. Face recognition using Laplacianfaces. *IEEE TPAMI*, 27(3):328–340, March 2005.
[12] Y. Saad. *Numerical Methods for Large Eigenvalue Problems*. Halstead Press, New York, 1992.

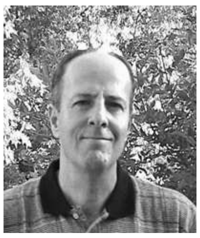
[13] D. B Graham and N. M Allinson. Characterizing Virtual Eigensignatures for General Purpose Face Recognition. *Face Recognition: From Theory to Applications*, 163:446–456, 1998.

[14] 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.

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

[16] 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.

Effrosyni Kokiopoulou received her Diploma in Engineering in June 2002, from the Computer Engineering and Informatics Department of the University of Patras, Greece. In June 2005, she received a Msc degree in Computer Science from the Computer Science and Engineering Department of the University of Minnesota, USA, under the supervision of prof. Yousef Saad. In September 2005, she joined the LTS4 Lab of the Signal Processing Institute, in the Swiss Federal Institute of Technology (EPFL), Lausanne, Switzerland. She is currently working towards her PhD degree under the supervision of prof. Pascal Frossard. Her research interests include multimedia data mining, machine learning, computer vision and numerical linear algebra. She is a student member of the IEEE.



Yousef Saad Yousef Saad is an Institute of Technology (I.T.) distinguished professor with the department of computer science and engineering at the University of Minnesota. He received the "Doctorat d'Etat" from the university of Grenoble (France) in 1983. He joined the university of Minnesota in 1990 as a Professor of computer science and a Fellow of the Minnesota Supercomputer Institute. He was head of the department of Computer Science and Engineering from January 1997 to June 2000, and became an IT distinguished professor in May 2005. From 1981 to 1990, he held positions at the University of California at Berkeley, Yale, the University of Illinois, and the Research Institute for Advanced Computer Science (RIACS). His current research interests include: numerical linear algebra, sparse matrix computations, iterative methods, parallel computing, numerical methods for electronic structure, and data analysis. He is the author of two (single authored) books and over 120 journal articles. He is also the developer or co-developer of several software packages for solving sparse linear systems of equations including SPARSKIT, pARMS, and ITSOL.