Enhanced graph-based dimensionality reduction with repulsion Laplaceans [☆]

E. Kokiopoulou[a], Y. Saad[b]

[a]*EPFL, LTS4 lab, Bat. ELE, Station 11; CH 1015 Lausanne; Switzerland. Email: effrosyni.kokiopoulou@epfl.ch*
[b]*Department of Computer Science and Engineering, University of Minnesota, Minneapolis, MN 55455. Email: saad@cs.umn.edu.*

Abstract

Graph-based methods for linear dimensionality reduction have recently attracted much attention and research efforts. The main goal of these methods is to preserve the properties of a graph representing the affinity between data points in local neighborhoods of the high dimensional space. It has been observed that, in general, supervised graph-methods outperform their unsupervised peers in various classification tasks. Supervised graphs are typically constructed by allowing two nodes to be adjacent only if they are of the same class. However, such graphs are oblivious to the proximity of data from different classes. In this paper, we propose a novel methodology which builds on 'repulsion graphs', i.e., graphs that model undesirable proximity between points. The main idea is to repel points from different classes that are close by in the input high dimensional space. The proposed methodology is generic and can be applied to any graph-based method for linear dimensionality reduction. We provide ample experimental evidence in the context of face recognition, which shows that the proposed methodology (i) offers significant performance improvement to various graph-based methods and (ii) outperforms existing solutions relying on repulsion forces.

Key words: Linear dimensionality reduction, orthogonal projections, supervised learning, face recognition, Graph Laplacean.

1. Introduction

The goal of dimensionality reduction techniques is to map high dimensional data samples to a lower dimensional space such that certain properties are preserved. Graph-based methods have attracted much research interest over the past few years. These methods typically rely on some graph to capture the salient geometric relations of the data in the high-dimensional space. This graph is usually called an affinity graph, since its edge set conveys some information about the proximity of the data in the input space. Once the affinity graph has been constructed, these methods derive the low dimensional samples by imposing that certain graph properties be preserved in the reduced space. This typically results in an optimization problem, whose solution provides the reduced data, or a mechanism to project data from the original space

[☆]Work supported by NSF under grants DMS 0810938 and DMS 0528492, and by the Minnesota Supercomputing Institute.

to low-dimensional space.

One may distinguish between two main categories of methods, depending on the graph property that is optimized. The first category optimizes the mapping by aiming to preserve data locality, i.e., by making points that are nearby in the input space also nearby in the reduced space. A few representative methods in this category include Locality Preserving Projections (LPP) [8] and Orthogonal Locality Preserving Projections (OLPP) [12, 11]. The second category of methods optimizes the mapping by aiming to preserve the geometry of the local neighborhoods, i.e.,the same weights which locally reconstruct data samples as convex combinations of near-by points should also reconstruct the corresponding points in the reduced space. One representative method in this category is Orthogonal Neighborhood Preserving Projections (ONPP) [12, 11].

In general, it has been observed that supervised graph-based methods outperform significantly their unsupervised peers in various recognition tasks. It is common practice to construct a supervised graph by only setting adjacent the nodes from the same class. The intuition is that during projection, when the data locality (or local geometry) is preserved, points from the same class will be mapped to points that are close by. This however, has one particular weakness; namely that points from different classes but nearby in some other measure (e.g., Euclidean distance) may be projected to points that are close-by in the low-dimensional space. This may lead to potential misclassification.

To remedy this weakness, we propose a methodology based on repulsion graphs. A repulsion graph is a graph whose edge set captures pairs of points that belong to different classes, but are close by in the input space. Maximizing the pairwise distances between these points will tend to repel these points from one another when they are projected. The proposed framework based on repulsion graphs is generic and can be applied to any graph-based method to improve its classification performance. The idea of repulsion forces, or negative energies, has been previously used in another context in graph-drawing techniques [13, 15] and in dimensionality reduction under different formulations (see e.g., [22]). We provide experimental evidence which shows that (i) including repulsion forces in various graph based methods can significantly boost their performance for face recognition and (ii) the proposed framework outperforms other competing solutions based on related repulsion ideas.

The rest of the paper is organized as follows. In Section 2 we review the general framework of graph based methods for dimensionality reduction. In particular, in Section 2.1 we discuss the different known means for building affinity graphs (supervised and unsupervised) and in 2.2 we revisit the various weighting schemes that are commonly used. Then, in Section 2.4 we review briefly the two main categories of methods from the literature. In the sequel, in Section 3 we introduce the proposed methodology on repulsion graphs. Next, in Section 4 we analyze the spectral properties of the involved matrix and in Section 5 we provide a physical interpretation of the repulsion forces in our framework. Finally, Section 6 presents our experimental results.

2. Linear dimensionality reduction: overview

Given a data set

$$X = [x_1, x_2, \dots, x_n] \in R^{m\times n}, \tag{1}$$

we wish to produce a set Y which is an accurate representation of X, but whose dimension d is much less than the original dimension 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.* Linear dimension reduction techniques replace the original data X by a matrix of the form

$$Y = V^\top X, \quad \text{where} \quad V \in R^{m\times d}. \tag{2}$$

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 an orthogonal matrix, then Y represents the orthogonal projection of X onto the V-space.

The best known technique in this category is Principal Component Analysis (PCA) which computes V such that the variance of the projected vectors is maximized. As is well-known, this leads to projecting the data onto the vector space spanned by the 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).

Recently developed graph-based techniques, start by building a graph to capture the local and global geometric structure of the data and then compute a mapping from high to low-dimensional space by imposing that certain graph properties in the reduced space are preserved. In what follows, we will discuss different ways in which graphs can be constructed and different objective functions used in a number of known methods and their variants.

2.1. Affinity graphs

Since the data samples are often high dimensional, it is common practice to use graphs in order to model their geometry and also to cope with the curse of dimensionality. Thus, we often use a graph

$$\mathcal{G} = (\mathcal{V}, \mathcal{E}) \tag{3}$$

whose nodes $\mathcal{V}$ correspond to the data samples and the edge set $\mathcal{E}$ models the relations between them. When we build the graph, depending on whether we use the class labels or not, we distinguish between two different cases: supervised and unsupervised.

Supervised case: the class graph.. Assume that we have c classes and that the data are given as in (1) along with their class labels $\ell : [1, \dots, n] \to [1, \dots, c]$. Here $\ell(i) = j$ means that the ith data sample belongs to the jth class. For the supervised case the class labels are used to build the graph. It has been observed in general that supervised methods perform better in many classification tasks relative to the unsupervised ones. The motivation here is to build the graph in a discriminant way to reflect the categorization of the data into different classes.

One simple approach is to build the data graph in (3) such that 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 which belong to the same class. Consider the structure of the induced adjacency matrix A. Let n_i be the number of samples which belong to the ith class. Observe that 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 adjacency matrix A will have a block diagonal form, where the size of the ith block is equal to the size n_i of the ith class. Hence, A will be of the following form,

$$A = \mathrm{diag}(A_1, A_2, \ldots, A_c).$$

In the above, the block A_i corresponds to the ith class.

The unsupervised case: neighborhood graphs.. In the unsupervised case, where class labels are not used, we typically define the edge set $\mathcal{E}$ in (3) in such a way that it captures the proximity of the data in high dimensional space. The k-NN graph is one very popular example that belongs to this category. In the k-NN graph two nodes x_i and x_j are made adjacent only if x_i is among the k nearest neighbors of x_j or vice versa. Another typical example is the ϵ-graph. In this case, a node x_i is made adjacent to all nodes $x_j, j \neq i$ that are within distance ϵ from it.

2.2. Graph weights

Edge weights are assigned in order to determine how each sample is influenced by its neighbors. This amounts to defining a weight matrix $W \in R^{n \times n}$ whose sparsity pattern is inherited by the adjacency matrix A. A few popular choices of weights are reviewed below.

Binary weights.. The weight matrix W is simply set equal to the adjacency matrix A.

Gaussian weights.. The weight matrix W is defined as follows.

$$W_{ij} = \begin{cases} e^{-\|x_i - x_j\|^2/t} & \text{if } A_{ij} \neq 0 \\ 0 & \text{otherwise} \end{cases}$$

These weights are also known as heat kernel weights. Note the presence of the parameter t.

Reconstruction weights,. These weights were first introduced in [16, 18]. The weight matrix in this case is built by computing optimal coefficients which relate a given point to its nearest neighbors in some locally optimal way. Each data sample along with its k nearest neighbors (approximately) are assumed to lie on a locally linear manifold. Hence, each data sample x_i is (approximately) reconstructed by a linear combination of its k nearest neighbors. The reconstruction errors are measured by minimizing the objective function

$$f(W) = \sum_i \left\| x_i - \sum_j W_{ij} x_j \right\|_2^2 . \tag{4}$$

The weights w_{ij} represent the linear coefficient for reconstructing the sample x_i from its neighbors $\{x_j\}$. For a fixed i, the weights w_{ij} are nonzero only when i and j are neighbors, and their sum is constrained to be equal to one. There is a simple closed-form expression for the weights. For details see [16, 18].

2.3. Graph Laplaceans

Graph Laplaceans provide one of the most common and useful tools for dimensionality reduction, see e.g. [3, 8, 12, 11, 16]. Let $\mathcal{N}(k)$ denote the adjacency list of vertex x_k. Then, a graph Laplacean is a matrix $L \in R^{n\times n}$, which has the following property,

$$L_{ij} \begin{cases} \leq & 0 \quad\quad\quad\quad\; \text{for} \quad j \in \mathcal{N}(i),\ j \neq i \\ = & -\sum_{k\neq i} L_{ik} \quad \text{if} \quad\;\; i = j. \end{cases}$$

For instance, observe that when W is a weight matrix, and if D is a diagonal matrix with $D_{ii} = \sum_j W_{ij}$, then $L = D - W$, is a graph Laplacean. The fundamental property of graph Laplaceans is that for any vector x of n scalars x_i, $i = 1, \dots, n$, we have:

$$x^\top Lx = \frac{1}{2}\sum_{ij} W_{ij}|x_i - x_j|^2.$$

This relation can be generalized to arrays with n column-vectors $y_i \in \mathcal{R}^m$ as follows

$$\text{Tr}\,[YLY^\top] = \frac{1}{2}\sum_{i,j=1}^{n} W_{ij}\|y_i - y_j\|_2^2 \tag{5}$$

where $Y \in \mathcal{R}^{m\times n}$ (see [12, 11]). Finally, note that graph Laplaceans have been used extensively for clustering, see e.g., [19], and for the closely related problem of graph partitioning [1]. The paper [21] gives a good overview of graph Laplaceans and their properties.

2.4. Preserving graph properties

So far we have seen different ways of building a graph to capture the geometric structure of the data. Imposing the property that low dimensional data set Y will preserve a certain graph property leads to an optimization problem, whose objective function is driven by this property. It is also common to include some constraints in this optimization problem and this leads to different methods. In what follows, we will discuss a few of these graph properties that are commonly used and the corresponding methods that they incur. For the ease of presentation, we will distinguish between two main categories of methods, depending on the objective function that they use.

Preserving locality. The main idea here is to project the data in the reduced space such that proximity is preserved. This implies that when two points x_i and x_j are close by in the input high dimensional space, then the corresponding points in the reduced space y_i and y_j should be positioned nearby as well. One way to achieve this is to use the following objective function, which involves the Laplacian matrix of the graph (see also (5)),

$$\Psi(Y) \equiv \text{Tr}\,[YLY^\top] = \frac{1}{2}\sum_{i,j=1}^{n} W_{ij}\|y_i - y_j\|_2^2. \tag{6}$$

Intuitively, when two points x_i and x_j are very similar, the corresponding weight W_{ij} will be large. Then, minimizing (6) will tend to force the distances $\|y_i - y_j\|_2^2$ to be small, i.e., it encourages points y_i and y_j to be placed close by in the low dimensional space.

A method in this class is that of *Laplacean Eigenmaps* [3], a method that is 'nonlinear' in the sense that the mapping between the original data (the x_i's) and the reduced data (the y_i's) is not an explicitly known linear mapping.

A linear relative of Laplacian eigenmaps is that of *Locality Preserving Projections* (LPP) [8]. Here, the mapping from X to the low-dimensional data is of the form $Y = V^\top X$ (see also (2)). LPP defines the projected points in the form $y_i = V^\top x_i$ by putting a high penalty for mapping nearest neighbor nodes. This is achieved by minimizing the same function (6) defined above, subject to the normalizing condition $YDY^\top = I$, where $D = diag(L)$. The column-vectors of the projector V are solutions of a generalized eigenvalue problem.

Finally, the method of *Orthogonal Locality Preserving Projections* (OLPP), introduced in [12, 11], is similar to LPP but it enforces orthogonality on the mapping ($V^\top V = I$) instead of the data ($YY^\top = I$ or $YDY^\top = I$). One of the key observations made (experimentally) in [12, 11] is that orthogonality of the mapping is quite important in face recognition. Note that this OLPP technique is distinct from one that is proposed in [4] and is referred to as Orthogonal Laplacean-Faces.

Preserving local geometry. The main idea here is to project the data in the reduced space such that local geometry is preserved. Recall that the reconstruction weights (Section 2.2) capture the geometric structure of local neighborhoods. Thus, one may impose that the projected data will be reconstructed by the same weights. This produces the following objective function,

$$\Phi(Y) = \sum_i \|y_i - \sum_j W_{ij} y_j\|_2^2 = \text{Tr}\left[Y(I - W^\top)(I - W)Y^\top\right]. \tag{7}$$

Observe that when the weights W_{ij} are fixed, then minimizing (7) will result in reduced data y_i's that are reconstructed from the same weights as their corresponding points x_i's in the input space.

A representative method that uses the above objective function is the *Locally Linear Embedding* (LLE) technique [16, 18]. LLE is a nonlinear method that seeks to preserve the intrinsic geometric properties of the local neighborhoods, by employing the objective function (7). The method of *Orthogonal Neighborhood Preserving Projections* (ONPP), introduced in [12, 11], imposes an explicit linear mapping from X to Y, which is in the form (2). ONPP minimizes the same objective function (7) as above, but now Y is restricted to being related to X by (2) with the additional constraint that the columns of V are orthonormal, i.e. $V^\top V = I$. The columns of the optimal V are eigenvectors associated with the d smallest eigenvalues of the matrix $\tilde{M} = X(I - W^\top)(I - W)X^\top$.

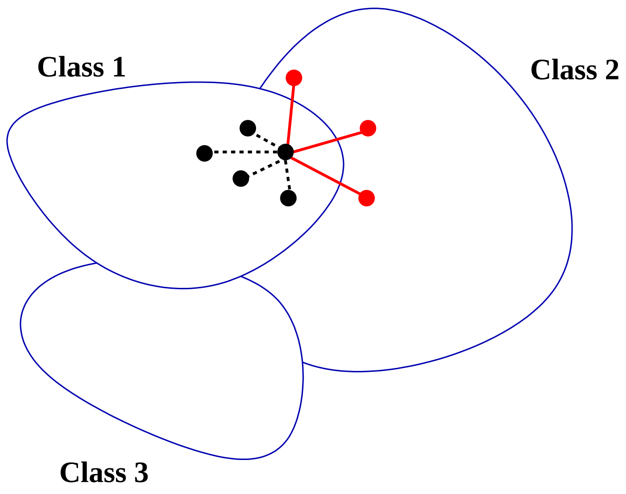


Figure 1: Illustration of the repulsion graph. The repulsion graph is obtained by only retaining among the k nearest neighbors of a node i, those nodes that are not in the same class as i (illustrated in red).

3. Repulsion graphs

The minimization of the objective functions (6) and (7) will yield points that are close by in the low-dimensional space, when they are close-by in the original space. Recall that in the supervised graph construction (Section 2.1) points that belong to the same class are made adjacent. So, in this case the data proximity is not used. Observe now that there may exist two points x_i and x_j that are close by, but do not belong to the same class. When a projection is performed, there is a risk that these two points which are close-by, will get projected to the same class, although they come from different classes. This situation can cause misclassification. In order to remedy this undiserable case, we introduce a methodology based on the concept of *repulsion graphs.*

A *repulsion graph* is one that is extracted from the k-NN graph, based on class label information. Our goal is to create a repulsion force between nearby points which are not from the same class. For example, when a k-nearest neighbor graph is used, a repulsion graph can be created by only *retaining among the k-nearest neighbors of a node i, those nodes that are not in the same class as i* (see Fig. 1). For simplicity, *we assume that the kNN graph is symmetrized by including the edge (j, i) whenever the edge (i, j) exists.* The weight matrix can be defined in the same way as was previously discussed in Section 2.2. In addition, as for standard Laplaceans,we also require that all row-sums be equal to zero. The key idea is that any objective function which will utilize the repulsion graph will tend to *maximize, rather than minimize* (6), where the Laplacian matrix now is associated with the repulsion graph. This 'repulsion Laplacean' will model a force - or rather an energy - which will tend to repel near-by points in different classes away from each other.

In the following discussion the class graph is used with either the Laplacean weights (OLPP) or the reconstruction weights (ONPP). This graph is only referenced as the class graph and requires no further notation. Its associated Laplacean is denoted by L. The repulsion graph is derived from a certain kNN graph, which we denote by $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. The repulsion graph itself is denoted by $\mathcal{G}^{(r)} = (\mathcal{V}^{(r)}, \mathcal{E}^{(r)})$, and its

Algorithm 1 The OLPP-R algorithm

```
1: Input:
     $X \in \mathbb{R}^{m\times n}$: training data.
     $k$: number of nearest neighbors.
     $\beta$: repulsion parameter.
     $d$ : dimension of the reduced space.
2: Output:
     $V \in \mathbb{R}^{m\times d}$: estimated projection matrix.
3: Build the supervised class graph and its associated Laplacian $L$.
4: Build the $k$-NN graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$.
5: Build the repulsion graph $\mathcal{G}^{(r)} = (\mathcal{V}^{(r)}, \mathcal{E}^{(r)})$ and its associated Laplacian $L^{(r)}$.
6: Compute the $d$ bottom eigenvectors $V = [v_1, \ldots, v_d]$ of $X(L - \beta L^{(r)})X^\top$.
```

associated Laplacean by $L^{(r)}$. Accordingly, the adjacency list for a given node i is now $\mathcal{N}^{(r)}(i)$. Assume for a moment that the weights are of the simple 'uniform' type, i.e., $l_{ij}^{(r)} = -1$ for $(i,j) \in \mathcal{E}^{(r)}$ and $i \neq j$ and $l_{ii}^{(r)} = -\sum_j l_{ij}^{(r)}$. In other words, the Laplacian matrix $L^{(r)}$ is derived from the weight matrix

$$W_{ij}^{(r)} = \begin{cases} 1 & \text{for} \quad (i,j) \in \mathcal{E},\ i \neq j, \quad \text{and } \ell(i) \neq \ell(j) \\ 0 & \text{otherwise} \end{cases} \tag{8}$$

by defining $L^{(r)} = D^{(r)} - W^{(r)}$, in which $D^{(r)}$ is the matrix of row-sums of $W^{(r)}$. This is a valid Laplacean as the row sums of the matrix are all zero, and the off-diagonal entries are non-positive. By the assumption of the indirection of the kNN graph (see above), $L^{(r)}$ is symmetric.

We will discuss in the sequel how the concept of repulsion graphs may be employed in the objective functions (6) and (7). We will consider the ONPP and OLPP methods as showcases but we should note that the methodology is generic and it is applicable to any graph-based method for dimensionality reduction. We will denote by ONPP-R (resp. OLPP-R) the ONPP (resp. OLPP) algorithms with repulsion.

ONPP-R. Recall that ONPP minimizes (7) under the constraints $Y = V^\top X$ and $V^\top V = I$. In order to introduce the repulsion force, we will add a term to the original objective function:

$$\Phi_\beta(Y) = \sum_i \|y_i - \sum_j w_{ij} y_j\|_2^2 - \beta \sum_i \sum_{j \in N^{(r)}(i)} \|y_i - y_j\|_2^2. \tag{9}$$

The first term in the above cost function uses the class graph as defined earlier in Section 2.1. We refer to the second term in the above expression as the *penalty term* and to the parameter β as the *penalty parameter*. If two projected entries x_i and x_j are not in the same class but they are close, then the edge (i,j) is part of the graph $\mathcal{G}^{(r)}$ and there is a penalty for having the two nodes close in Y. Due to the negative sign, the penalty term will tend to be larger in absolute value in order to minimize the objective function.

One may think that it is more natural to divide the first term of the right-hand side of (9) by the second, in order to avoid having an additional parameter (β). However, this would yield a non-orthogonal projector and the resulting technique does not work as well as a few experiments will confirm.

Relations (6) and (7) show that the above objective function can be expressed as

$$\Phi_\beta(V) = \mathrm{Tr}\left[V^\top X(M - \beta L^{(r)})X^\top V\right], \tag{10}$$

where $M = (I - W^\top)(I - W)$. If we impose orthogonality constraints i.e., $V^\top V = I$, then V is obtained from the d bottom eigenvectors of matrix $X(M - \beta L^{(r)})X^\top$.

OLPP-R. Introducing the repulsion force into OLPP can be achieved in a very similar way – essentially all that is needed is to replace the ONPP matrix M by the Laplacean matrix L associated with the class graph, which was discussed in Sections 2.1 and 2.3. The above objective function to minimize can be expressed as

$$\Psi_\beta(V) = \mathrm{Tr}\left[V^\top X(L - \beta L^{(r)})X^\top V\right]. \tag{11}$$

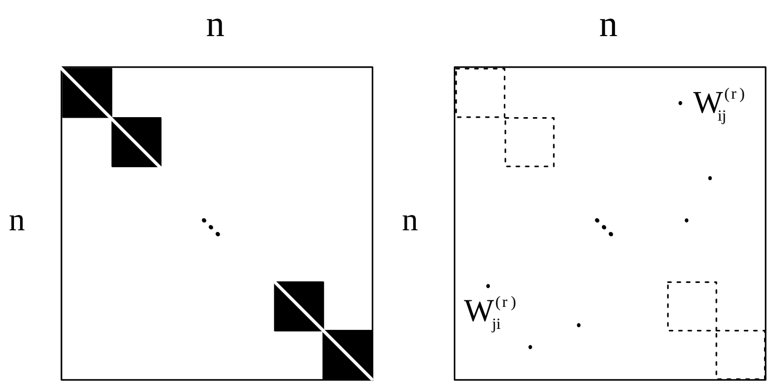
Similarly, imposing orthogonality constraints on V, the solution is obtained from the d bottom eigenvectors of matrix $X(L-\beta L^{(r)})X^\top$. We call the matrix $L-\beta L^{(r)}$ as the *augmented repulsion matrix* and we will study its spectral properties in Section 4 which follows. The main steps of OLPP-R are illustrated in Algorithm 1. A similar algorithm for ONPP-R can be constructed by replacing $L - \beta L^{(r)}$ by $M - \beta L^{(r)}$.

Alternative weights.. Alternative weights can be used for the repulsion graph to account for the fact that it may be better to use a higher penalty weight for those points not in the same class that are closest in high-dimensional space. The following often gives markedly better results than constant weights:

$$W_{ij}^{(r)} = \frac{1}{\sigma + \frac{\|x_i - x_j\|^2}{\|x_i\|_2^2 + \|x_j\|_2^2}} \tag{12}$$

One may question the use of this alternative in view of the fact that it requires an additional parameter. Our observation, is that the resulting performance is not too overly sensitive on the choice of σ and so it is beneficial to use the above weight in practice.

Discussion.. Note in passing that ideas related to repulsion graphs have already been exploited in the literature. For example, they are extensively used in graph drawing as a tool to faithfully represent a graph in two dimensions while avoiding 'clutter' [13, 15, 5]. Repulsion graphs are actually used with different goals in graph-drawing. The model discussed in [15] aims at creating forces which will tend to attract points to each other if they are alike (same class in our case) and repel points from each other in the opposite case. In the seminal paper by Frucheterman and Reingold the stated goal is to "(i) draw adjacent vertices near each other but (ii) not too close to each other". The intuition for these models is rooted in physical n-body systems. However, these methods utilize as a guiding principle to: *make adjacent nodes attract each other but make all points repel each other.* This is in slight contrast with our technique whose goal is to make adjacent nodes (alike points) attract each other but make only a subset of points repel each other.



(a) The diagonal of W is zero.

(b) The dashed line indicates the zero diagonal blocks of $W^{(r)}$.

Figure 2: Structure of weight matrices W (left panel) and $W^{(r)}$ (right panel).

Note finally that our definition of the repulsion graphs is flexible and can accommodate scenarios with limited supervision. This is typical for instance in semi-supervised learning, where only a few samples are labelled and the majority of data samples are unlabeled. Another example of limited supervision is the scenario where instead of class labels, only a few pairs of similar and dissimilar samples are provided. In both cases, one may adapt the repulsion graph accordingly to exploit the available supervision information.

4. Characterization of the augmented repulsion matrix

In this section we characterize the spectral properties of the augmented repulsion matrix involved in cost function (11):

$$A_\beta = L - \beta L^{(r)} \ . \tag{13}$$

When ordered by class, this matrix will be a sparse matrix with diagonal blocks corresponding to the different classes (due to L), and some off diagonal entries which represent near-by points which are not in the same class (due to $L^{(r)}$). Consider first the class-graph Laplacean L. According to Section 2.1, the class graph consists of c cliques each associated with a class. The weight assigned to each edge in the clique is $w_{ij} = 1/n_l$ where n_l is cardinality of class number l. Let z_l be the indicator vector for class number l, i.e., the n-dimensional vector with entries equal to one for indices belonging to class l and zero elsewhere. Furthermore, denote by $I(z_l)$ the matrix diag(z_l), where, using matlab notation, diag(x) represents the diagonal matrix with diagonal entries $x_1, x_2, \cdots, x_n$. With this, the weight matrix is the sum of weight matrices associated with each class:

$$W = \sum_{l=1}^{c} W_l, \quad \text{with} \quad W_l = \frac{1}{n_l}\left[z_l z_l^\top - I(z_l)\right] . \tag{14}$$

Each of the W_l matrices has a zero diagonal, and constant nonzero entries in locations (i, j) when i and j have the same label (see Fig. 2(a)). The sum on the row entries of W are equal to $(1 - 1/n_l)$ for a row in class l. Using this, it can be verified that the associated Laplacean is

$$L = I - \sum_{l=1}^{c} \frac{1}{n_l} z_l z_l^\top.$$

Since the set of vectors $\{z_l/\sqrt{n_l}\}$ forms an orthonormal system, the matrix L is the orthogonal projector onto the space orthogonal to the span of $z_1, z_2, \cdots, z_c$. It has exactly $n-c$ eigenvalues equal to one and c zero eigenvalues.

Consider now the whole matrix (13). We will need to make an assumption about the repulsive Laplacean, namely that it is in the form

$$L^{(r)} = I - W^{(r)}, \tag{15}$$

where $W^{(r)}$ is the weight matrix. This means that we scale the weights (8) so they sum to one for each vertex. The resulting Laplacean is often called a "normalized Graph Laplacean" [21]. Note that $W^{(r)}$ is not positive definite and so $\lambda_{min}(W^{(r)}) \le 0$. We also denote by Z the orthonormal $n \times c$ matrix with column-vectors $z_l/\sqrt{n_l}$, $l = 1, \cdots, c$. With this A_β becomes,

$$A_\beta = (1-\beta)I + \beta W^{(r)} - ZZ^\top.$$

Using well-known results we can say something about the eigenvalues of A_β.

Theorem 4.1. *Assume the weight matrices for the class graph and repulsion graph satisfy assumtions(14) and (15) respectively, and label all eigenvalues increasingly. Then for $j > c$ the eigenvalues of A_β satisfy the inequalities,*

$$(1-\beta) + \beta\lambda_{j-c}(W^{(r)}) \le \lambda_j(A_\beta) \le (1-\beta) + \beta\lambda_j(W^{(r)}) \tag{16}$$

In particular, if $\beta \le 1/\lambda_{max}(L^{(r)})$ then A_β has at most c negative eigenvalues.

Proof. The proof is based on Weyl's theorem, see [10, sec. 4.3], which essentially states that for two $n \times n$ Hermitian matrices A and B we have: $\lambda_{k+j-n}(A+B) \le \lambda_j(A) + \lambda_k(B)$ whenever the indices $k+j-n$, k and j are 'valid' i.e., between 1 and n. A second part of the theorem states that $\lambda_j(A) + \lambda_k(B) \le \lambda_{k+j-1}(A+B)$ when the indices $k+j-1$, k and j are 'valid'. Apply first the first part of the theorem to the decomposition

$$A_\beta + ZZ^\top = (1-\beta)I + \beta W^{(r)}. \tag{17}$$

Here A_β is used as the A of the theorem and $ZZ^\top$ as the B matrix. Note that the first $n-c$ eigenvalues of $ZZ^\top$ are all zero and the last c ones are equal to one. Taking $k = n-c$ yields the first part of inequality (16)

$$\lambda_{j-c}\left[(1-\beta)I + \beta W^{(r)}\right] \le \lambda_j(A_\beta) + 0 \to (1-\beta) + \beta\lambda_{j-c}(W^{(r)}) \le \lambda_j(A_\beta) \ .$$

For the second part of the inequality we use the second part of Weyl's theorem to the same decomposition (17) with $k=1$, which yields

$$\lambda_j(A_\beta) + \lambda_k(ZZ^\top) \le \lambda_{k+j-1}\left[(1-\beta)I + \beta W^{(r)}\right] \to \lambda_j(A_\beta) \le (1-\beta) + \beta\lambda_j(W^{(r)}) \ .$$

Finally, take $j = c+1$ in the left part of (16): $1 - \beta + \beta\lambda_1(W^{(r)}) \leq \lambda_{c+1}(A_\beta)$. Note that $\lambda_1(W^{(r)}) \leq 0$ as observed before. Then $1 - \beta + \beta\lambda_1(W^{(r)}) \geq 0$ iff $1/\beta - [1 - \lambda_1(W^{(r)})] \geq 0$, iff $\beta \leq 1/(1 - \lambda_1(W^{(r)}))$. In this situation $\lambda_{c+1}(A_\beta) \geq 0$. From (15), note finally that $1 - \lambda_{min}(W^{(r)})$ is the largest eigenvalue of $L^{(r)}$. ∎

5. Physical interpretation of repulsion forces

In this section, we provide a physical interpretation of the repulsion forces as they are instantiated in our framework. In order to avoid confusion with class labels we denote by $z(k)$ the k component of vector z in the following discussion. For a vector z of 2-norm unity, the quantity $\frac{1}{2}z^\top A_\beta z$ represents an energy for the vector which has components $z(k)$ in item k of the set. The method seeks the directions of lowest energies. There are actually two parts to the energy: the term $\frac{1}{2}z^\top Lz$ which represents the class-related energy and the term $\beta\frac{1}{2}z^\top L^{(r)}z$ which is the repulsion energy. A negative energy is one for which the repulsion part dominates the class-related energy.

Denoting the Rayleigh quotient $z^\top A_\beta z/(z^\top z)$ by $\mu(z)$, the gradient of energy is $\nabla A_\beta z = A_\beta z - \mu(z)z$. This vector contains components of forces acting on each vertex for the particular distribution z. This reaches a zero value for any eigenvector of A_β, which means that eigenvectors yield stationary distributions.

Consider now one of the indicator vectors z_i and note that z_i represents simply the centroid of class i (when written in the basis $x_1, ..., x_n$). Note that these are the stationary vectors for the case when $\beta = 0$, i.e., for the classical method without repulsive Laplacean, and so it is instructive study how the zero (stationary) forces change when $\beta > 0$. First, we have

$$A_\beta z_i = (I - ZZ^\top)z_i - \beta(I - W^{(r)})z_i = -\beta z_i + \beta W^{(r)}z_i \ . \tag{18}$$

One can exploit the interesting structure of the weight matrix $W^{(r)}$. When $w_{kj} > 0$ i.e., when k and j are in the same class, then $\ell(k) = \ell(j)$ and so $w_{kj}^{(r)} = 0$. Therefore, the terms w_{kj} and $w_{kj}^{(r)}$) cannot be nonzero at the same time. In other words, when ordered by class, the matrix $W^{(r)}$ has zero diagonal blocks in entries related to the same class i (see Fig. 2(b)). In particular, it is easy to see that the vector $W^{(r)}z_i$ has nonzero components only outside of the set i. Then it results from (18) that $z_i^\top W^{(r)}z_i = 0$ and so (18) implies that $\mu(z_i) = -\beta$. Then the force associated with z_i is

$$f_i = A_\beta z_i - \mu(z_i)z_i = -\beta z_i - \beta W^{(r)}z_i + \beta z_i = \beta W^{(r)}z_i \ .$$

This vector represents the average of all the columns of $W^{(r)}$ associated with the class i. Each vertex ν which is in a different class from class i will have a component

$$\frac{\beta}{n_i} \sum_{j \, \in \, \mathcal{N}^{(r)}(\nu)} w_{j\nu}^{(r)} .$$

The net effect is that *there is positive force (repulsive) away from the centroid of class i, on each of the vertices ν linked to class i by an edge in the repulsive graph.* All other nodes have a zero force acting on them.

6. Experimental Results

In this section we evaluate the impact of repulsion forces to the performance of dimensionality reduction methods for face recognition. Again, we will consider ONPP and OLPP for illustration purposes, although we should bear in mind that the idea of repulsion graphs is generic and can be employed to improve the performance of several other graph-based methods as well. In particular, we will compare the two novel methods ONPP-R, and OLPP-R (see Section 3), with their predecessors as well as with other methods. The comparisons are with the basic PCA method and a few other methods tested in [12, 11]. These include the Laplacian-faces algorithm [9] ONPP and OLPP [12, 11] and Fisherfaces [2]. The implementations of these methods are the same as those in [12, 11] with a few exceptions which will be specified later.

6.1. Experimental setup

We use 4 data-sets which are publicly available for the experiments. These are the UMIST [7], the ORL [17], the AR set [14] and the Essex dataset [20]. After cropping and downsampling, each facial image was represented lexicographically as a high dimensional vector. 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. To reduce the likelihood of bias 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.

For ONPP, OLPP, and Laplacean Faces, the *supervised graphs* were employed (see Section 2.1). The two modified versions use the same penalty parameter in all cases which is $\beta = 0.2$. However, the matrices $X\tilde{M}X^\top$ and $XLX^\top$ are normalized by their traces so that in effect they are scaled to have unit trace. In all experiments, the kNN-graph was built using k=15 nearest neighbors. Recall that the number of neighbors in the repulsion graph is not known in advance, since it depends on class information. A sigma of the order of 10.0 in (12) has yielded good results and will be used throughout the experiments of this section.

Note that the Laplaceanfaces method uses Gaussian weights. A procedure for determining a good value for the width σ of the Gaussian envelope was employed. The procedure first samples 1000 points randomly and then it computes the pairwise distances among them. Then σ is set to half the median of those pairwise distances. This gives a good and reasonable estimate for the value of σ.

Finally, we should mention that the above methods have been pre-processed with a preliminary PCA projection step. Typically an initial PCA projection is used in order to reduce the dimensionality of the data vectors to $n - c$. This was used in [12, 11]. In reality we found that we can use a dimension that is much less than $n - c$ without affecting the final results. Specifically, we employ the Lanczos algorithm [6]. If d is the desired subspace dimension for the reduced space, we let the Lanczos algorithm run for $K * d$ steps, where K is a parameter (set to $K = 12$ in the experiments) and select the eigenvectors associated with its half largest eigenvalues for the pre-projection step. This is much less expensive than performing a full-fledged SVD and gave very similar results.

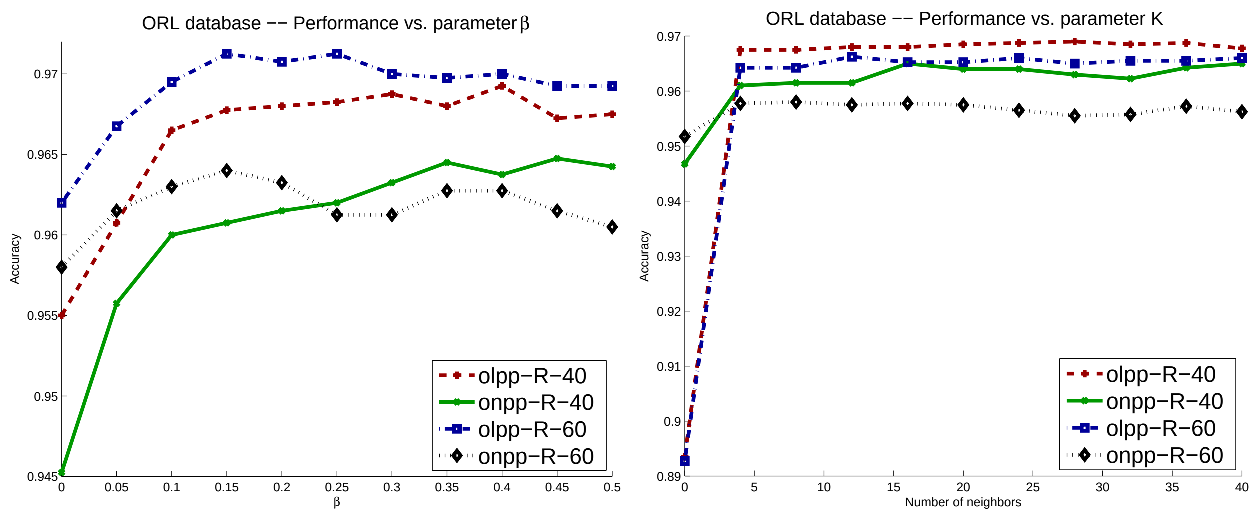


Figure 3: Performance of ONPP-R and OLPP-R for varying β (left side) and K (right-side). Two cases are shown for each plot, one with 40 eigenvectors and the other with 60.

6.2. Selection of parameters

There are two main parameters in the modified versions of ONPP and OLPP. The first is the same parameter that is used in all kNN-graph based methods – namely the parameter k which determines the number of neighbors. The second is the penalty parameter β. First, we will illustrate the behavior of the methods with respect to the penalty parameter. This is done with the ORL test set. We fix the number of training images to 5 per subject and let all other images be test images. The number of eigenvectors is set to 40 and then to 60 (two separate experiments). We vary β with the values 0.05:0.05:4 (matlab notation). Figure 3 plots the performance of each of the two methods OLPP-R and ONPP-R with respect to β. The number of tests is 20 per each value of β.

The main observation from this test and many others performed is that beyond a certain value of β the performance varies very little. Note that for example when $d = 40$, for $\beta > 0.1$ the errors vary only in the 3rd digits and hover around 0.96 for both OLPP-R and ONPP-R.

Though it may be possible to develop practical means of extracting optimal values of β, this experiment and others seem to suggest that the gains from such an optimal value will be limited. We generally select $\beta = 0.2$.

We performed a similar experience in which we vary the number of neighbors used to define the kNN graph. The particular case of $k = 0$ yields the base methods again since the repulsion graph is empty in this situation. As can be seen the variation of performance with respect to K is even less pronounced than it is with respect to β. Starting with $k = 10$, the performance curves are essentially constant for the $d = 60$ case and show a very small change for $d = 40$.

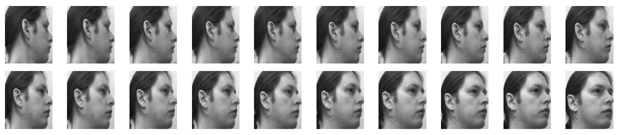
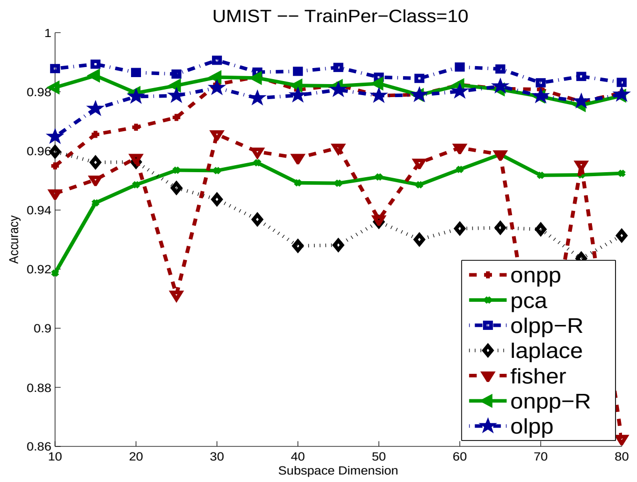


Figure 4: Sample face images from the UMIST database. The number of different poses poses for each subject is varying.

6.3. UMIST

The UMIST database [7] 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 publicly available from S. Roweis' web page[1]. Figure 4 illustrates a sample subject from the UMIST database along with its first 20 views. For this experiment, 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 5. The best error rate achieved by each method and the corresponding dimension d of the reduced space are also reported in the figure.

The most significant observations from the plot are the following. First, the modified versions outperform the non-modified ones. The improvement is most notable for low-dimensions. In fact, the second observation is the remarkable performance of these methods for low dimensions. In particular, it appears that the best possible performance is reached much earlier and that it does not deteriorate much for larger dimensions. This 'stable' feature is very appealing. In contrast, the other methods reach their best performance for larger values of d.



Best achieved error rates

Method	Dim	Error
onpp	35	1.49e-02
pca	65	4.12e-02
olpp-R	30	9.33e-03
laplace	10	4.03e-02
fisher	30	3.44e-02
onpp-R	15	1.45e-02
olpp	65	1.81e-02

Figure 5: Accuracy reached versus d the reduced space dimension d for the UMIST database. The table on the right shows the best errors achieved by each method and the corresponding dimensions.

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

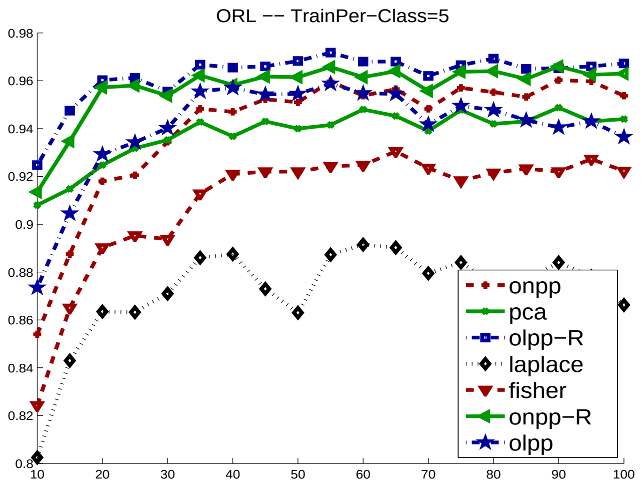
6.4. ORL

The ORL (formerly Olivetti) database [17] is a small set which contains 40 individuals and 10 different images for each individual including variation in facial expression (smiling/non smiling) and pose. Figure 6 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 expressions/poses per subject and use the remaining 5 as a test set. We experiment with the dimension of the reduced space $d = [15 : 5 : 100]$ and for each value of d we compute the average error rate across 20 random realizations of the training set.

Figure 6: Sample face images from the ORL database. There are 10 available facial expressions and poses for each subject.

Figure 7 illustrates the results. The observations made earlier for the test with the UMIST are still valid. Note in particular, the remarkable performance of OLPP-R and ONPP-R for low dimensions. The best possible performance is reached much earlier and that it does not deteriorate much for larger dimensions. The other methods reach their best performance for larger values of d (larger than 100) and accuracy declines thereafter.

Notice also that the modified version of OLPP is just slightly better that the modified version of ONPP. The best error rates achieved by each method are also reported in the figure.



Best achieved error rates

Method	Dim	Error
onpp	90	3.97e-02
pca	90	5.12e-02
olpp-R	55	2.82e-02
laplace	60	1.08e-01
fisher	65	6.95e-02
onpp-R	90	3.40e-02
olpp	55	4.12e-02

Figure 7: Error rate with respect to the reduced dimension d for the ORL database. The table on the right shows the best errors achieved by each method and the corresponding dimensions.

6.5. AR

We use a subset of the AR face database [14] which contains 126 subjects under 8 different facial expressions and variable lighting conditions for each individual. Figure 8 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 $d = [10 : 5 : 100]$.

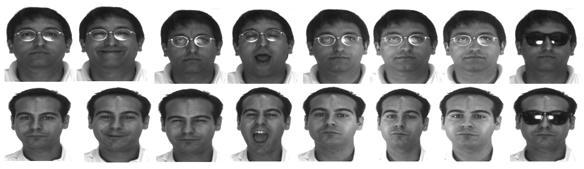
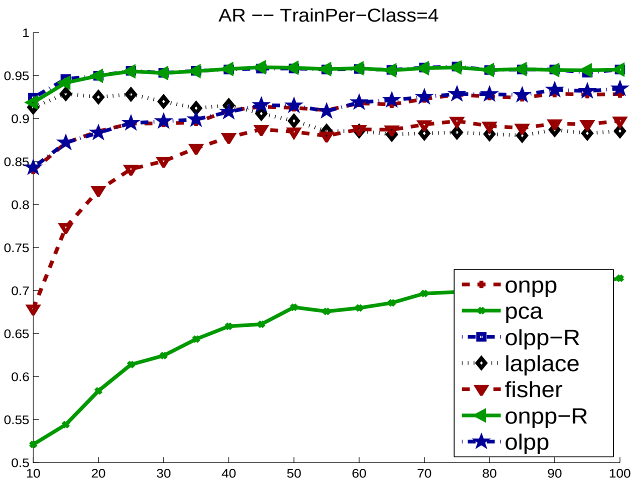


Figure 8: Sample face images from the AR database.

The results are illustrated in Figure 9. Once again we observe that ONPP-R and OLPP-R outperform the other methods across all values of d. Finally, observe that for this database, PCA does not perform too well. In addition, OLPP-R and ONPP-R yield very similar performances for this case.

It was observed in [12, 11] that orthogonality of the columns of the dimensionality reduction matrix V is very important for data visualization and recognition purposes. The experiments of this paper confirm this observation. We tested other formulations of the repulsion modified eigenvalue problems and we observed experimentally that orthogonality of the columns of V is crucial for the recognition performance.



Best achieved error rates

Method	Dim	Error
onpp	90	7.10e-02
pca	100	2.86e-01
olpp-R	75	3.96e-02
laplace	15	7.12e-02
fisher	100	1.03e-01
onpp-R	45	4.04e-02
olpp	100	6.53e-02

Figure 9: Error rate with respect to the reduced dimension d for the AR database. The table on the right shows the best errors achieved by each method and the corresponding dimensions.

6.6. University of Essex data set

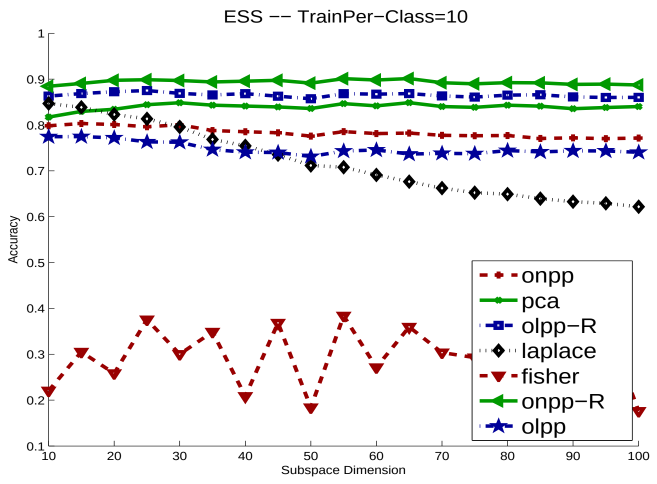
This collection is a available from http://cswww.essex.ac.uk/mv/allfaces/index.html. The whole collection contains a total of 7900 images of 395 individuals and is organized in 4 different sets – from easy to hard: 'faces94', 'faces95', 'faces96', and 'grimaces'. We opted to consider the collection faces95 which contains faces of 72 individuals. A sequence of 20 images per individual [2] was taken using a fixed camera, while the subject

[2]One or more of the subjects had 19 images only.

takes one step forward towards the camera. This introduces significant head variations between images of the same subject. Each picture of the faces95 set is a 200×180 RGB image, i.e., 36000 pixels per image. We cropped each picture slightly by removing the top 20 rows, 10 bottom rows and 20 columns of pixels from each side. Then we downsampled the picture by a factor of 4 in each direction. This yielded pictures of size 36×43 (or 1548 pixels per image), which were then converted to grayscale pictures. Two random samples of 10 images from the data set are shown in Figure 10.

Figure 10: Sample face images from the Essex/faces95 database. Both samples are selected randomly among the 1440 images of the set.

This set is harder than other ones tested so far. This is due to the rich variation of poses and illumination. A similar experiment as earlier was performed. Again we used half of the images (10) from each subject to form the training and the other images to form the test set. Results with 7 methods are reported in Figure 11. Note that PCA does quite well on this set while LDA does very poorly. Only ONPP-R achieves an error rate that is lower than 10%.



Best achieved error rates

Method	Dim	Error
onpp	15	1.97e-01
pca	65	1.51e-01
olpp-R	25	1.25e-01
laplace	10	1.53e-01
fisher	55	6.16e-01
onpp-R	65	9.88e-02
olpp	15	2.25e-01

Figure 11: Accuracy reached versus d the reduced space dimension d for the faces95 subset of the Essex database. The table on the right shows the best errors achieved by each method and the corresponding dimensions.

6.7. Other forms of repulsion graphs

We have selected one way of defining a repulsion graph to model repulsive forces but there are others. We have mentioned one option in Section 3 which uses a different objective function, namely to define the objective function to be the ratio rather than the difference of the attractive and repulsive energies as in

(9). As was mentioned, this would give a non-orthogonal projector and does not perform as well as the experiments reveal.

Another option is a related technique discussed in [22] and called "Discriminant Neighborhood Embedding" (DNE). The main difference between DNE and the technique of this paper is that DNE does not use the class graph to define the attraction forces. Instead it relies entirely on the kNN graph to model both the attraction part and repulsion parts. Hence, it falls short of exploiting the full supervision information, since those points that belong to the same class and are not neighbors, are not used in DNE. Specifically, DNE uses the following weights on the knn graph

$$F_{ij} = \begin{cases} +1 & (x_i \in knn(j) \vee x_j \in knn(i)) \wedge (c_i = c_j), \\ -1 & (x_i \in knn(j) \vee x_j \in knn(i)) \wedge (c_i \neq c_j), \\ 0 & \text{otherwise} \end{cases} \qquad (19)$$

With this defined, the method described in [22] prescribes to miniminze an objective function of the form $\mathrm{Tr}\,[V^\top(D-F)V]$ where D is defined as is usually done to obtain a proper Laplacian with zero row-sums. Once this is done, *only those eigenvectors associated with negative eigenvalues are selected to define the projector*. No explanation was given for this choice. Since the number of eigenvectors is not known in advance and can be quite large, we simply took the lowest d eigenvalues as in other similar approaches.

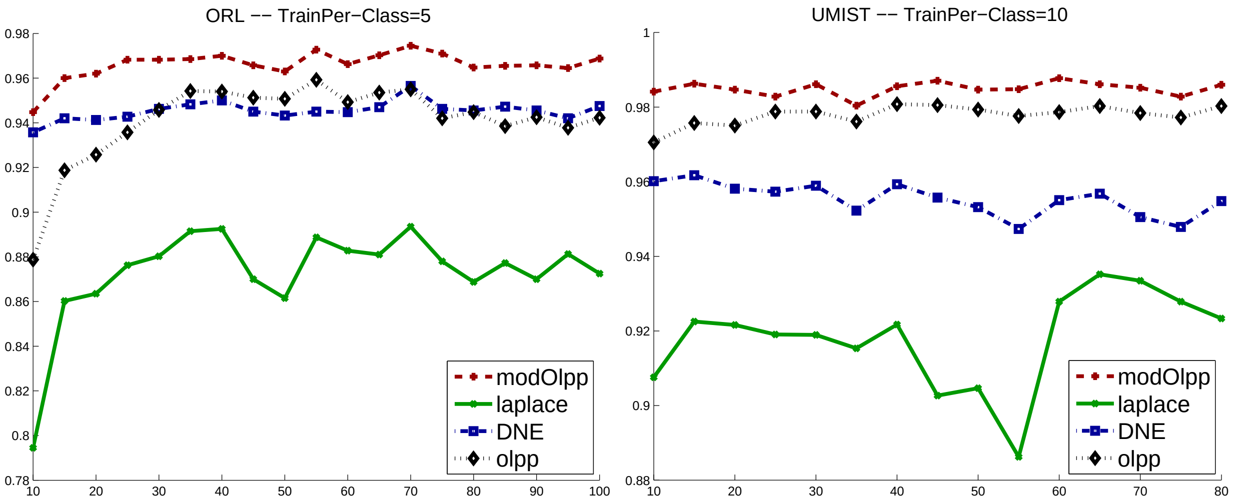


Figure 12: A comparison with DNE [22], a method which does not use the class graph. Left: ORL set, Right: UMIST.

The experiments we performed showed that this technique does not perform as well. We show two sample experiments in Figure 12 with the datasets ORL and UMIST. Here we only test 4 methods each time: the standard OLPP and its modified version with repulsion graphs, the Laplacian eigenmaps approach, and the DNE method discussed above. We use a kNN graph with $k = 15$ in all cases. The plot shows that while DNE is an improvement over Laplacian faces, it is not competitive with OLPP-R. In fact for these examples its (best achieved) performance is even worse than that of the standard OLPP. This is representative of a general pattern observed with this method. In addition, the variant which uses only negative eigenvalues gave similar if not somewhat worse results.

7. Conclusion

We have proposed a methodology based on repulsion forces to enhance the graph-based methods for dimensionality reduction. The main idea is to repel points from different classes that are nearby in the input space. We have presented a generic approach of introducing repulsion forces into recent state-of-the-art algorithms, by adding a repulsion matrix to the graph Laplacean or affinity matrix. We have shown that this helps improve the performance of graph based techniques in a significant way. While the new methods demand two additional parameters, namely β (penalty coefficient) and k (number of nearest neighbors in kNN graph), experience shows that the performance of the methods are not overly sensitive to the values of these parameters when they are taken in a certain range.

References

[1] Stephen T. Barnard, Alex Pothen, and Horst Simon. A spectral algorithm for envelope reduction of sparse matrices. *Numerical Linear Algebra with Applications*, 2(4):317–334, 1995.

[2] 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.

[3] M. Belkin and P. Niyogi. Laplacian eigenmaps and spectral techniques for embedding and clustering, 2002.

[4] D. Cai, X. He, J. Han, and H.-J. Zhang. Orthogonal Laplacianfaces for face recognition. *IEEE Trans. on Image Processing*, 15(11):3608–3614, 2006.

[5] T. M. J. Fruchterman and E. M. Reingold. Graph drawing by force-directed placement. *Software - Practice and Experience*, 21(11):1129–1164, 1991.

[6] G. H. Golub and C. F. Van Loan. *Matrix Computations*. Johns Hopkins University Press, Baltimore, MD, 3rd edition, 1996.

[7] 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.

[8] X. He and P. Niyogi. Locality preserving projections. *In Proc. Conf. Advances in Neural Information Processing Systems*, 2003.

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

[10] R. A. Horn and C. R. Johnson. *Matrix Analysis*. Cambridge University Press, Cambridge, 1985.

[11] E. Kokiopoulou and Y. Saad. Orthogonal neighborhood preserving projections. In J. Han et al., editor, *IEEE 5th Int. Conf. on Data Mining (ICDM05), Houston, TX, Nov. 27-30th*, pages 234–241. IEEE, 2005.

[12] E. Kokiopoulou and Y. Saad. Orthogonal neighborhood preserving projections: A projection-based dimensionality reduction technique. *IEEE TPAMI*, 29:2143–2156, 2007.

[13] Yehuda Koren. On spectral graph drawing. In *In COCOON 03, volume 2697 of LNCS*, pages 496–508. Springer-Verlag, 2003.

[14] A. M. Martinez and R. Benavente. The AR face database. Technical Report 24, CVC, 1998.

[15] Andreas Noack. An energy model for visual graph clustering. In *Proceedings of the 11th International Symposium on Graph Drawing (GD 2003), LNCS 2912*, pages 425–436. Springer-Verlag, 2004.

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

[17] 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.

[18] L. Saul and S. Roweis. Think globally, fit locally: unsupervised learning of nonlinear manifolds. *Journal of Machine Learning Research*, 4:119–155, 2003.

[19] J. Shi and J. Malik. Normalized cuts and image segmentation. In *Proc. IEEE Int. Conf. Computer Vision*, pages 731–737, 1997.

[20] L. Spacek. University of essex face database, 2002. http://cswww.essex.ac.uk/mv/allfaces/index.html.

[21] Ulrike von Luxburg. A tutorial on spectral clustering. Technical Report TR-149, Max-Planck für biologische Kybernetik, Tuebingen, Germany, 2006.

[22] Wei Zhang, Xiangyang Xue, Hong Lu, and Yue-Fei Guo. Discriminant neighborhood embedding for classification. *Pattern Recogn.*, 39(11):2240–2243, 2006.