Embedding New Observations via Sparse-Coding for Non-linear Manifold Learning

B. Raducanu[1] and F. Dornaika[2,3]

[1] *Computer Vision Center, Barcelona, SPAIN*
[2] *Department of Computer Science and Artificial Intelligence University of the Basque Country UPV/EHU, San Sebastian, SPAIN*
[3] *IKERBASQUE, Basque Foundation for Science, Bilbao, SPAIN*

Abstract

Non-linear dimensionality reduction techniques are affected by two critical aspects: (i) the design of the adjacency graphs, and (ii) the embedding of new test data—the out-of-sample problem. For the first aspect, the proposed solutions, in general, were heuristically driven. For the second aspect, the difficulty resides in finding an accurate mapping that transfers unseen data samples into an existing manifold. Past works addressing these two aspects were heavily parametric in the sense that the optimal performance is only achieved for a suitable parameter choice that should be known in advance.

In this paper, we demonstrate that Sparse representation theory not only serves for automatic graph construction as shown in recent works, but also represents an accurate alternative for out-of-sample embedding. Considering for a case study the Laplacian Eigenmaps, we applied our method to the face recognition problem. To evaluate the effectiveness of the proposed out-of-sample embedding, experiments are conducted using the k-nearest neighbor (KNN) and Kernel Support Vector Machines (KSVM) classifiers on six public

face datasets. The experimental results show that the proposed model is able to achieve high categorization effectiveness as well as high consistency with non-linear embeddings/manifolds obtained in batch modes.
Keywords: Non-linear manifold learning, out-of-sample embedding, sparse representation, face recognition

---

1. Introduction

Manifold learning refers to the problem of recovering the structure of a manifold from a set of unordered sample data. Manifold learning is often equated with dimensionality reduction, where the goal is to find an embedding or unrolling of the manifold into a lower dimensional space such as certain relationships between samples are preserved. Such embeddings are typically used for visualization. In recent years, a new family of non-linear dimensionality reduction techniques for manifold learning has emerged. The most known are: Kernel Principal Component Analysis (KPCA) [1], Locally Linear Embedding (LLE) [2, 3], Isomap [4], Supervised Isomap [5], Laplacian Eigenmaps (LE)[6, 7]. This family of non-linear embedding techniques appeared as an alternative to their linear counterparts which suffer severe limitation when dealing with real-world data: i) they assume the data lie in an Euclidean space and ii) they may fail to get a faithful representation of data distribution when the number of samples is too small. On the other hand, the non-linear dimensionality techniques are able to discover the intrinsic data structure by exploiting the local topology. In general, they attempt to optimally preserve the local geometry around each data sample while using the rest of the samples to preserve the global structure of the data.

The non-linear methods such as Locally Linear Embedding (LLE), Laplacian Eigenmaps, Isomap, Hessian LLE (hLLE) [8] focus on preserving the local structure of data. LLE formulates the manifold learning problem as a neighborhood-preserving embedding, which learns the global structure by exploiting the local linear reconstructions. It estimates the reconstruction coefficients by minimizing the reconstruction error of the set of all local neighborhoods in the dataset. Isomap extends the classical Multidimensional Scaling (MDS) [9] by computing the pairwise distances in the geodesic space of the manifold. Essentially, Isomap attempts to preserve geodesic distances when data are embedded in the new low dimensional space. Based on the spectral decomposition of the Laplacian of a graph, Laplacian Eigenmaps actually try to find Laplacian eigenfunction on the manifold. Maximum Variance Unfolding (MVU) [10] is a global algorithm for nonlinear dimensionality reduction, in which all the data pairs, nearby and far, are considered. MVU attempts to 'unfold' a dataset by pulling the input patterns as far apart as possible subject to constraints that distances and angles between neighboring points are strictly preserved.

The main issues of the non-linear methods are: (1) the quality of embedded space is very sensitive to the choice of free parameters used in the data graph construction [11, 12], and (2) they do not provide an explicit mapping function between low and high dimensional spaces [13, 14]. Such function is essential for ensuring the continuity of low dimensional representation and projecting data between spaces. Many existing manifold learning techniques do not naturally contain an out-of-sample extension, so research has been undertaken to find ways of extending manifold learning techniques to handle

new samples. The out-of-sample extension problem has not received much attention by researchers since it was considered a pure non-linear regression problem [15, 16]. Therefore, the out-of-sample problem has been addressed quite satisfactorily by applying Radial Basis Function networks in order to approximate the optimal mapping function [15]. However, the quality of Radial Basis Function networks relies on the careful selection of a few parameters which are chosen empirically [17, 18]. In [19], the author presented an algorithm, Locally Smooth Manifold Learning, for learning the structure of a manifold in terms of tangent vectors. Rather than pose manifold learning as the problem of recovering an embedding, they posed the problem in terms of learning a warping function for traversing the manifold using the learned tangent vectors. Smoothness assumptions on this warp allowed the method generalize to unseen data.

In [20], the authors cast MDS, ISOMAP, LLE, and LE in a common framework, in which these methods are seen as learning eigenfunctions of a kernel. The authors try to generalize the dimensionality reduction results for the unseen data samples. In [21], the author proposes a method based on probabilistic mixtures of factor analyzers to 1) model the density of images sampled from such manifolds and 2) recover global parameterizations of the manifold. A globally nonlinear probabilistic two-way mapping between coordinates on the manifold and images is estimated by combining several, locally valid, linear mappings. In [22], the authors propose a novel solution which involves approximating the kernel eigenfunction using Gaussian basis functions. They also show how the width of the Gaussian can be tuned to achieve extrapolation. Their method was applied to Maximum Variance Un-

folding (MVU) method [10]. In [23], the proposed method works by learning the transformation that maps the neighborhood of the unlearnt sample from the high to the low-dimensional space. This transformation is then applied to the new sample to obtain an estimation of its low-dimensional embedding.

In this paper, we address the out-of-sample extension problem. We adopt the sparse representation approach as an optimal solution to the 'out-of-sample' problem. The sparse representation was recently used as an effective alternative to the parametric construction of the adjacency graph [12]. Without any loss of generality, we chose the Laplacian Eigenmaps as one of the non-linear dimensionality reduction techniques to test our method. We present a generalized out-of-sample extension solution using the recent findings in sparse coding theory. Unlike existing approaches we do not require information to be retained from the learning process, such as the pairwise distance matrix or the resultant eigenvectors, we simply learn the mapping from the original high-dimensional data and its low-dimensional counterpart. Although the proposed method integrates the locality preserving principle in its derivation, it is intended to be independent of any specific manifold learning algorithm.

The paper is structured as follows. In section 2, we briefly review the Laplacian Eigenmaps as well as the L_1 graph construction. In section 3, we introduce our proposed approach for the out-of-sample problem based on sparse representation. Section 4 contains the experimental results performed on six face datasets. We evaluate the performance of the proposed out-of-sample method for the face recognition problem. Finally, in section 5 we present our conclusions.

2. Background

2.1. Review of Laplacian Eigenmaps

Laplacian Eigenmaps is a recent non-linear dimensionality reduction technique that aims to preserve the local structure of data [6]. Using the notion of the Laplacian of a graph, this non-supervised algorithm computes a low-dimensional representation of the dataset by optimally preserving local neighborhood information in a certain sense. We assume that we have a set of N samples $\{\mathbf{x}_i\}_{i=1}^N \subset \mathbb{R}^D$. The original LE starts with building a graph on the data samples. In this graph, the nodes represent the data samples and the edges quantify the similarity among pairs of samples. There are several ways for setting the edges of the graph. For instance, the most common strategy is to use a K-nearest-neighbor or ϵ-ball graph, or a full mesh (all pairs are connected). Once the edges are set, one can weigh each edge $\mathbf{x}_i \sim \mathbf{x}_j$ by a symmetric affinity function $W_{ij} = K(\mathbf{x}_i; \mathbf{x}_j)$, typically Gaussian:

$$W_{ij} = \exp(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{\beta}) \tag{1}$$

where β is a suitable positive scalar. It is usually set to the average of squared distances between all pairs.

LE seeks latent points $\{\mathbf{y}_i\}_{i=1}^N \subset \mathbb{R}^L$ that minimize $\frac{1}{2}\sum_{i,j} \|\mathbf{y}_i - \mathbf{y}_j\|^2 W_{ij}$, which discourages placing far apart latent points that correspond to similar observed points. If $\mathbf{W} \equiv W_{ij}$ denotes the symmetric affinity matrix and $\mathbf{D}$ is the diagonal weight matrix, whose entries are column (or row, since $\mathbf{W}$ is symmetric) sums of $\mathbf{W}$, then the Laplacian matrix is given $\mathbf{L} = \mathbf{D} - \mathbf{W}$. The

objective function can also be written as:

$$\frac{1}{2}\sum_{i,j}\|\mathbf{y}_i-\mathbf{y}_j\|^2 W_{ij} = tr(\mathbf{Z}^T\,\mathbf{L}\,\mathbf{Z}) \tag{2}$$

where $\mathbf{Z}^T = \mathbf{Y} = [\mathbf{y}_1,\ldots,\mathbf{y}_N]$ is the $L \times N$ matrix of embedded data and $tr(.)$ denotes the trace of a matrix. The i^{th} row of the matrix $\mathbf{Z}$ provides the vector $\mathbf{y}_i$—the embedding coordinates of the sample $\mathbf{x}_i$.

The matrix $\mathbf{Z}$ (or equivalently $\mathbf{Y}$) is the solution of the optimization problem:

$$\min_{\mathbf{Z}} tr(\mathbf{Z}^T\,\mathbf{L}\,\mathbf{Z})\ \ s.t.\ \ \mathbf{Z}^T\,\mathbf{D}\,\mathbf{Z} = \mathbf{I},\ \ \mathbf{Z}^T\,\mathbf{L}\,\mathbf{1} = \mathbf{0} \tag{3}$$

where $\mathbf{I}$ is the identity matrix and $\mathbf{1} = (1,\ldots,1)^T$. The first constraint eliminates the trivial solution $\mathbf{Z} = \mathbf{0}$ (by setting an arbitrary scale) and the second constraint eliminates the trivial solution $\mathbf{1}$ (all samples are mapped to the same point). Standard methods show that the embedding matrix is provided by the matrix of eigenvectors corresponding to the smallest eigenvalues of the generalized eigenvector problem,

$$\mathbf{L}\,\mathbf{z} = \lambda\,\mathbf{D}\,\mathbf{z} \tag{4}$$

Let the column vectors $\mathbf{z}_0,\ldots,\mathbf{z}_{N-1}$ be the solutions of (4), ordered according to their eigenvalues, $\lambda_0 = 0 \leq \lambda_1 \leq \ldots \leq \lambda_{N-1}$. The eigenvector corresponding to eigenvalue 0 is left out and only the next eigenvectors for embedding are used. The embedding of the original samples is given by the row vectors of the matrix $\mathbf{Z}$, that is, $\mathbf{Y} = [\mathbf{y}_1,\mathbf{y}_2,\ldots,\mathbf{y}_N] = \mathbf{Z}^T$.

$$\mathbf{x}_i \longrightarrow \mathbf{y}_i = (z_1(i),\ldots,z_L(i))^T \tag{5}$$

where $L < N$ is the dimension of the new space.

From equation (4), we can observe that the dimensionality of the subspace obtained by LE is limited by the number of samples N.

2.2. Review of L1 graph construction

In traditional graph construction process, the graph adjacency structure and the graph weights are derived separately (Previous section). In [12], the authors argue that the graph adjacency structure and the graph weights are interrelated and should not be separated. Thus it is desired to develop a procedure which can simultaneously carry out these two tasks within one step. Indeed, many experiments show that the performance of classification tasks in the embedded space of LE obtained with a traditional graph construction scheme can highly depend on the choice of the neighborhood size in the constructed graph [24, 25, 26]. Choosing the ideal size in advance can be a very difficult task.

The basic idea of [12] is to simultaneously estimate the graph adjacency structure and graph weights. To this end, every sample image is coded as a sparse linear combination of the rest of the training samples [27, 28]. This is carried out by implementing an L_1 minimization process to obtain the sparse representation of that sample as a linear combination of the remaining training samples. The obtained sparse coefficients will reflect the relation among samples [29, 30], and hence they will provide the graph adjacency structure as well as the weights of its edges where the absolute value of a coefficient can be considered as the edge weight.

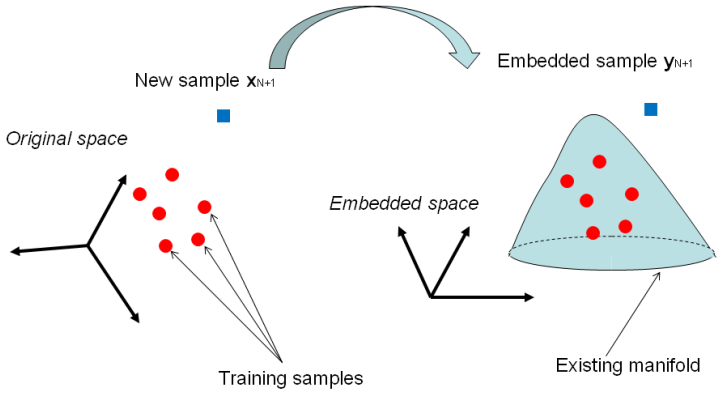

Figure 1: The out-of-sample problem consists in finding the embedding coordinate od a newly unseen sample.

3. Proposed out-of-sample embedding

In this section, we show that the theory of sparse representation (coding) can be used for solving the out-of-sample extension problem without relying on traditional heuristics that are usually parametric. For a case study, we use the Laplacian Eigenmaps for the non-linear embedding. The reason of our choice is motivated by the fact that this transform is widely used by the machine learning community for spectral clustering [31, 32, 33].

3.1. Projection of new samples

Assume we have obtained an LE embedding $\mathbf{Y}_s = (\mathbf{y}_1, \ldots, \mathbf{y}_N)$ of seen samples $\mathbf{X}_s = (\mathbf{x}_1, \ldots, \mathbf{x}_N)$ and consider an unseen sample (out-of-sample observation) in observed space $\mathbf{x}_{N+1}$ (See Figure 1). The natural way to em-

bed the new sample would be to recompute the whole embedding $(\mathbf{Y}_s, \mathbf{y}_{N+1})$ for $(\mathbf{X}_s, \mathbf{x}_{N+1})$ using Eq. (3). This is computationally costly and does not lead to define a mapping for new samples; we seek a way of keeping the old embedding fixed and embed new sample based on that. Then, the next step is to recompute the embedding while keeping the old embedded samples fixed and impose that the embedding of the new sample (vector $\mathbf{y}_{N+1}$) should minimize the following target function:

$$\sum_{i=1}^{N} \|\mathbf{y}_{N+1} - \mathbf{y}_i\|^2 W_{(N+1)i} \tag{6}$$

$$= \sum_{i=1}^{N} (\mathbf{y}_{N+1} - \mathbf{y}_i)^T (\mathbf{y}_{N+1} - \mathbf{y}_i) W_{(N+1)i} \tag{7}$$

The above should correspond to a minimum, and thus the derivative with respect to $\mathbf{y}_{N+1}$ of the target function should disappear:

$$2 \sum_{i=1}^{N} (\mathbf{y}_{N+1} - \mathbf{y}_i) W_{(N+1)i} = \mathbf{0} \tag{8}$$

From the above, we can conclude that the embedding $\mathbf{y}_{N+1}$ is given by:

$$\mathbf{y}_{N+1} = \frac{\sum_{i=1}^{N} W_{(N+1)i}\, \mathbf{y}_i}{\sum_{i=1}^{N} W_{(N+1)i}} \tag{9}$$

The above formula stipulates that the embedding of an unseen sample is simply the linear combination of all fixed embedded samples where the linear coefficients are set to the similarities between the unseen sample and the existing samples.

Whenever $W_{(N+1)i}$ is set to a Kernel function (i.e., $W_{(N+1)i} = K(\mathbf{x}_{N+1}, \mathbf{x}_i)$, Eq. (9) is equivalent to the Laplacian Eigenmaps Latent Variable Model (LELVM) introduced in [34].

3.2. Computation of the similarity coefficients via Sparse Representation

The problem of out-of-sample embedding is reduced to the estimation of the similarities $W_{(N+1)i}, i = 1, \ldots, N$. In [34], these $W_{(N+1)i}$ were computed using a K nearest neighbor and a Heat Kernel. However, it is well known that the neighborhood size as well as the Kernel parameter may affect the embedding process. We will bypass this limitation by using the coding provided by sparse representation.

We apply the sparse coding/representation principle for computing the set of coefficients $W_{(N+1)i}$ [30, 35]. Let the vector $\mathbf{a} = (W_{(N+1)1}, W_{(N+1)2}, \ldots, W_{(N+1)N})^T$. Thus, the objective is to compute the vector $\mathbf{a}$ given the unseen sample $\mathbf{x}_{N+1}$ and the training data $\mathbf{X}$. Based on a linear coding, one can assume that the following equation is approximately satisfied:

$$\mathbf{x}_{N+1} = \sum_{i=1}^{N} a_i \, \mathbf{x}_i = \mathbf{X}\,\mathbf{a}$$

The sparse solution is given by solving the following L_1 minimization problem:

$$\min_{\mathbf{a}} \|\mathbf{a}\|_{L_1} \quad s.t. \quad \mathbf{x}_{N+1} = \mathbf{X}\,\mathbf{a} \tag{10}$$

As suggested in [28], in many practical cases, data are corrupted by large errors. Thus, the above formulation should be modified. The unseen sample can be given by:

$$\mathbf{x}_{N+1} = \sum_{i=1}^{N} a_i \, \mathbf{x}_i + \mathbf{e} = \mathbf{X}\,\mathbf{a} + \mathbf{e} \tag{11}$$

where $\mathbf{e}$ is a vector of errors—a fraction of its entries are nonzero. The nonzero entries of $\mathbf{e}$ model which elements or pixels in $\mathbf{x}_{N+1}$ are corrupted

or occluded. The locations of corruption can differ for different test samples and are not known in advance. The errors may have arbitrary magnitude and therefore cannot be ignored or treated with techniques designed for small noise such as the one given in Eq. (10).

The goal is to minimize the L_1 norm of the vector $\mathbf{a}$ as well as that of $\mathbf{e}$:

$$\min_{\mathbf{a},\mathbf{e}} (\|\mathbf{a}\|_{L_1} + \|\mathbf{e}\|_{L_1}) \quad s.t. \quad \mathbf{x}_{N+1} = \mathbf{X}\,\mathbf{a} + \mathbf{e} \tag{12}$$

Let $\mathbf{a}'$ denote the vector $\mathbf{a}' = (\mathbf{a}^T, \mathbf{e}^T)^T$ and $\mathbf{I}$ denote the $D \times D$ identity matrix, then the objective function (12) can be written as:

$$\min \|\mathbf{a}'\|_{L_1} \quad s.t. \quad [\mathbf{X}\ \mathbf{I}]\,\mathbf{a}' = \mathbf{x}_{N+1} \tag{13}$$

Moreover, one can assume that the corrupting error $\mathbf{e}$ has a sparse representation with respect to some basis $\mathbf{A}_e$. That is, $\mathbf{e} = \mathbf{A}_e\,\mathbf{u}_0$ for some sparse vector $\mathbf{u}_0$. Here, we have chosen the special case $\mathbf{A}_e = \mathbf{I}$ as $\mathbf{e}$ is assumed to be sparse with respect to the natural pixel coordinates. If the error $\mathbf{e}$ is know to be more sparse with respect to another basis, e.g., Fourier or Haar, one can simply replace the identity matrix by another matrix $\mathbf{A}_e$.

Although no sparse priors are imposed, the sparse property of the coefficient vector $\mathbf{a}$ is naturally generated by the L_1 optimization. The optimization of (13) is carried out using the matlab package provided by [36].

Once the vector $(\mathbf{a}^T, \mathbf{e}^T)^T$ is computed, the similarity coefficients $W_{(N+1)i}$ are set to:

$$W_{(N+1)i} = |a_i|, i = 1, \ldots, N$$

3.3. Advantages of the proposed out-of-sample embedding scheme

Although our proposed out-of-sample formula (Eq. (9)) is similar to that of the Latent Variable Model [34], it has two interesting differences and advantages:

1. For the LVM scheme, the neighborhood size must be set manually, and the optimal setting may be different for different datasets. In our scheme, the computation of similarity coefficients adapts to the dataset through the use of sparse coding. No parameter is required.
2. There have been many ways to compute the similarity coefficients and the most popular one among them is the typical Heat Kernel (Gaussian weighting function) described in Eq.(1). However, the Gaussian aperture may affect the final classification results significantly, and how to optimally determine this parameter is still an open problem. Our scheme get rid of this since we exploit the sparseness property of the deduced coefficients in order to express both adjacency structure and the associated weights without any predefined parameter.

4. Performance evaluation

To validate the effectiveness of our proposed approach, we applied it to the face recognition problem. The experimental results are reported in terms of recognition accuracy and a similarity measure of the embedding ('out-of-sample' vs. 'batch-mode').

4.1. Datasets

In our experiments, we considered six public face datasets, which are characterized by a large variation in face appearance.

1. **Yale**[1]**:** The YALE face dataset contains 165 images of 15 persons. Each individual has 11 images. The images demonstrate variations in lighting condition, facial expression. Each image is resized to 32×32 pixels.
2. **ORL**[2]**:**. There are 10 images for each of the 40 human subjects, which were taken at different times, varying lighting, facial expressions (open/closed eyes, smiling/not smiling) and facial details (glasses/no glasses). The images were taken with a tolerance for some tilting and rotation of the face up to 20^o.
3. **UMIST**[3]**:**. The UMIST dataset contains 575 gray images of 20 different people. The images depict variations in head pose.
4. **Extended Yale - part B**[4]**:**. It contains 16128 images of 28 human subjects under 9 poses and 64 illumination conditions. In our study, a subset of 1800 images has been used. Figure 5 shows some face samples in the extended Yale Face Database B.
5. **PF01** It contains the true-color face images of 103 people, 53 men and 50 women, representing 17 different images (1 normal face, 4 illumination variations, 8 pose variations, 4 expression variations) per person. All the people in the dataset are Asians. There are three kinds of sys-

[1] $http://see.xidian.edu.cn/vipsl/database\_Face.html$

[2] $http://www.cl.cam.ac.uk/research/dtg/attarchive/facedatabase.html$

[3] $http://www.shef.ac.uk/eee/research/vie/research/face.html$

[4] $http://vision.ucsd.edu/\sim leekc/ExtYaleDatabase/ExtYaleB.html$

tematic variations, such as illumination, pose, and expression variations in the dataset.

6. **PIE**[5]**:** We use a reduced dataset containing 1926 face images of 68 individuals. The images contain poses variations, illumination variations, and facial expression variations. The image size is 32×32 pixels with 256-bit grey scale.

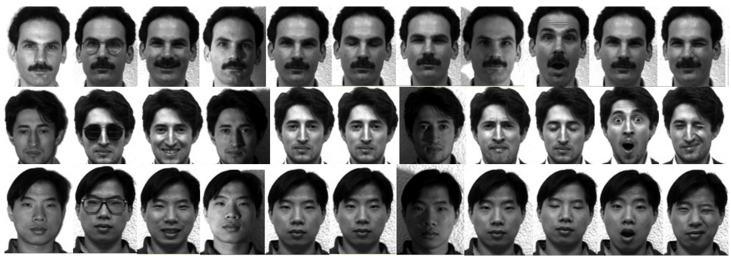

Figure 2: Some samples in Yale dataset.

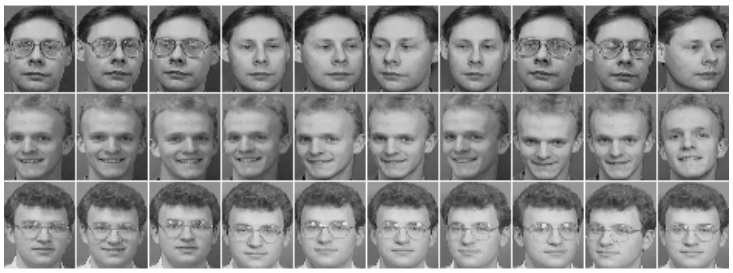

Figure 3: Some samples in ORL dataset.

4.2. Recognition accuracy

To make the computation of the embedding process more efficient, the dimensionality of the original face samples was reduced by applying random

[5] *http : //www.ri.cmu.edu/projects/project_*418*.html*

Figure 4: Some samples in UMIST dataset.

Figure 5: Some samples in Extended Yale dataset.

Figure 6: Some samples in PF01 dataset.

Figure 7: Some samples in PIE dataset.

projections [37]. It has a similar role to that of PCA yet with the obvious advantage that random projections do not need any training data.

We have compared our method with three other approaches. The first method is is the Latent Variable Model (LVM), proposed in [34]. The second one is a linearization of the existing mapping $\mathbf{X}_s \rightarrow \mathbf{Y}_s$. To this end, we use a simple linear regression in order to infer a matrix transform $\mathbf{A}$ that best approximates the existing mapping through the linear equation $\mathbf{Y}_s = \mathbf{A}^T \mathbf{X}_s$. We stress the fact that the linearization has not been thoroughly tested as an an out-of-sample method. Instead, this linearization was used for spectral regression (e.g., [38]). The third method is a representation based on Radial-Basis Functions (RBF) [17, 15]. In our implementation of the RBF method, we use Gaussian kernels whose number is equal to the number of training samples. In other words, we consider each training sample as a center. The aperture of the Gaussian kernels was set to the average squared distances between the pairs of the training samples.

For each face dataset and for every embedding method, we conducted three groups of experiments for which the percentage of training samples was set to 30%, 50% and 70% of the whole dataset. The remaining data was used for testing. Here, the testing implies: (i) the out-of-sample embedding of the unseen observation (face) **(new observation embedding)**, and (ii) assigning it a class-label through the use of a classifier in the embedded space **(recognition)**.

We considered for comparison two classifiers: nearest neighbor (NN) and Support Vector Machines (SVM). For a given out-of-sample embedding method, the recognition rate was computed for several dimensions belonging

to the interval $[5, L_{max}]$, where L_{max} is a parameter directly related with the number of training samples. In Figures 8 and 10 we depict the recognition rate (based on NN and SVM, respectively) as a function of dimension of the embedded space, considering 30% of data for training, for all the 4 out-of-sample embedding methods. The curves have been obtained by averaging the results over ten random splits. In the case of NN classifer, we use 1 neighbor for classification. Regarding SVM, we use a gaussian kernel. Similar results are depicted in Figures 9 and 11, but this time considering 70% of data for training.

In Tables 1 and 2, we present the best (average) performance obtained by each 'out-of-sample' method, based on 10 random splits using NN and SVM, respectively. Numbers in bold designate the best results. For the case of LVM method, the ϵ parameter corresponds to the number of neighbors used to approximate the unseen sample. We could appreciate that the smaller this number is, the better the result of LVM method.

From the results, we can draw the following conclusions:

(i) For the case of the NN classifer, the above results confirm the superiority of our approach when compared with existing ones. We can observe that this superiority was obtained for all datasets and for all dimensions tested for the obtained embedding space. We can also observe that the linearization method provided the poorest results, which can be explained by the fact that the linear method is global and does not take into account the local adjacency information. We can also appreciate that, for the NN classifier, the performance of LVM and RBF depends on the dataset used. There is no general trend that shows that one method is better than the other.

(ii) For the case of SVM, the sparse representation does not guarantee always the best recognition accuracy rate, but it can be outperformed by the RBF method in some cases. This could be explained by the fact that both RBF and SVM are highly non-linear techniques which can benefit each other well. For the SVM classifier, we can observe that the superiority of RBF was only obtained for a few cases and for high dimensions (see the PF01 and PIE datasets in Figures 10 and 11). If we consider the PF01 dataset with 70% of data for training (the lower part of Figure 11), we can observe that the RBF method provided better results than the sparse representation method for dimensions that are larger than 550. This dimension becomes 845 for the PIE dataset (the lower part of Figure 11). In practice, it should be a trade-off between a high recognition rate and a compact representation with a reduced number of dimensions. Thus, this requirement favors again our proposed sparse representation method since it has the best performance for low dimensions even when challenging face datasets (such as PF01 and PIE) are considered. It is worth mentioning that this advantage is not shown in the tables since the latter depict the best performances over the number of dimensions.

4.3. Assessing manifold reconstruction accuracy

In the previous section, we have evaluated the recognition performance of the proposed out-of-sample embedding method. However, the main role of the out-of-sample embedding method is to complete the reconstruction of the manifold in the embedded space (i.e., by adding the new observations in the embedded space). To this end, we can compare the coordinates of the new embedded observations with their coordinates computed in the batch

mode. The batch mode assumes that the whole dataset is used in order to get the non-linear manifold learning.

In order to quantify the accuracy of the out-of-sample embedding methods, we use the following error measure:

$$e = \frac{dist(\mathbf{Y}, \hat{\mathbf{Y}})}{\|\hat{\mathbf{Y}}\|_F}$$

where $dist(,)$ denotes the Procrustes distance [39], $\|\mathbf{A}\|_F$ denotes the Frobenius norm of the matrix $\mathbf{A}$, and $\mathbf{Y}$ and $\hat{\mathbf{Y}}$ are the test data that are provided by the out-of-sample method and the associated batch one, respectively. The above error can quantify the dissimilarity between the batch mode geometric configuration and the out-of-sample geometric configuration related to the test observations.

In Table 3 we show some results based on this definition for all the modalities and for all out-of-sample methods. Numbers marked in bold represent the best alignment between 'out-of-sample' and 'batch-mode' embedding. The smaller the number, the better the alignment. We could conclude that our proposed sparse representation method offers the best similarity with the batch mode embedding

Additionally, Figure 12 shows, for the UMIST dataset (in the modality 70-30), the evolution of the dissimilarity obtained by the out-of sample methods as function of dimensionality of the embedded space. We could appreciate that, for all out-of-sample methods, the dissimilarity distance is decreasing with the increase of the embedding dimensionality. However, after a certain value, the sparse representation method, again, guarantees the highest similarity. We can also observe that the alignment obtained by the

LVM and our proposed method is much better than that of the linearization and RBF methods.

4.4. Assessing algorithms' complexity

Regarding the algorithms' complexity, the critical aspect is represented by the computational load needed to project a new sample on the embedded space. Let d denote the adopted dimensionality of the non-linear embedded space. Let D denote the sample dimension in input space and N denote the number of training samples. The Linearization method is based on a linear regression. This out-of-sample method requires: (i) the computation of the pseudo-inverse of a $D \times N$ matrix, (ii) a matrix multiplication to get the linear transform (a $d \times D$ matrix), and (iii) a matrix-vector product to obtain the projection of the unseen sample. On its turn, the RBF method requires: (i) computing $N(N+1)/2$ elements of a symmetric Kernel matrix associated with the training set, (ii) computing N Kernel elements (test sample with the whole training samples), (iii) computing the inverse of an $N \times N$ matrix, and (iv) a matrix-vector product to obtain the projection of the unseen sample. Regarding the Latent Variable Model, this method requires: (i) the estimation of the K nearest neighbors, (ii) the computation of the Kernel responses between the test sample and these K neighbors, (iii) a weighted sum of K embedded samples (Eq. (9)). Our proposed Sparse Representation method consists of two main steps: (i) Computing the blending weights via an L_1 minimization (Eq. (13)), and (ii) Performing a weighted sum of the embedded samples (Eq. (9)). It is obvious that the first step is the most computationally expensive one [36]. Its complexity depends on the size of the

matrix used as a dictionary for L_1 coding (the dictionary refers to the basis matrix $[\mathbf{X}, \mathbf{I}]$ in Eq. (13)) which is given by $D \times (N + D)$. In order to quantify the computational complexity of all out-of-sample embedding methods used in this paper, we have considered the Extended Yale dataset, in the modality 70-30 (1241 samples for training).

Table 4 illustrates the CPU times (in milliseconds) required to project a new sample. The dimensionality of the data in the non-linear manifold was set to 1200 (this is related to the size of the training set). We performed the experiments using a non-optimized MATLAB code running on a PC equipped with a dual-core Intel processor at 2 Ghz and 2 Gb of RAM memory. It should be noted that the proposed method has the highest computational load since it relies on an L_1 minimization based on a very large dictionary. Despite of the increased time required by the proposed Sparse Representation method, this should not be considered as a serious drawback. Indeed, the above CPU time was obtained with a large dictionary formed by 1241 training samples, each having 200 elements (each image is projected using Random Projection adopting 200 axes). Therefore, alternative techniques that allow the use of a small dictionary will help us to achieve a more efficient implementation (e.g., see Figure 14). These techniques can rely on online learning and clustering (in both sequential and chunk update modalities).

Besides this, we also performed two additional studies (for Sparse Representation only). One aims to estimate the computational complexity of projecting a new sample, as function of the dimensionality of the input data. As we have mentioned before, in our case, due to the use of Random Projections, we have reduced the dimensionality of the input image to 200. In

Table 5 we measured the computation time at several dimensions, starting with 80 and finishing with 200, with a step of 20. For a more convenient visualization of the complexity evolution, the same results were represented as a plot in Figure 13. As it can be seen, by reducing the dimension of samples in the dictionary to half, the CPU time is decreased by 23%. The other study estimates the CPU time as a function of the size of the training data. For a fixed dimensionality of the input data (in this case 100), we have considered several sizes of training set, ranging from 600 up to 1200 with an incremental step of 100. The results are shown in Table 6 and the corresponding plot is depicted in Figure 14. As can be seen by reducing the size of the training set to half, the CPU time decreases by 73.8%. Due to the rapid increase of the computational complexity with the size of the training set, we are considering the possibility to adopt an L_1 minimization strategy based on a representative subset of the training set only.

5. Conclusions and Future Work

In this paper, we demonstrated that sparse representation can serve as an accurate alternative for out-of-sample embedding. Considering for a case study the Laplacian Eigenmaps, we applied our method to the face recognition problem. Indeed, the proposed out-of-sample embedding in general provided the best classification accuracy as well as the best alignment between out-of-sample mode and batch mode. The experimental results demonstrate that our algorithm can maintain an accurate low-dimensional representation of the data without any parameter tuning. A natural extension of our approach is its application to online learning and incremental embedding.

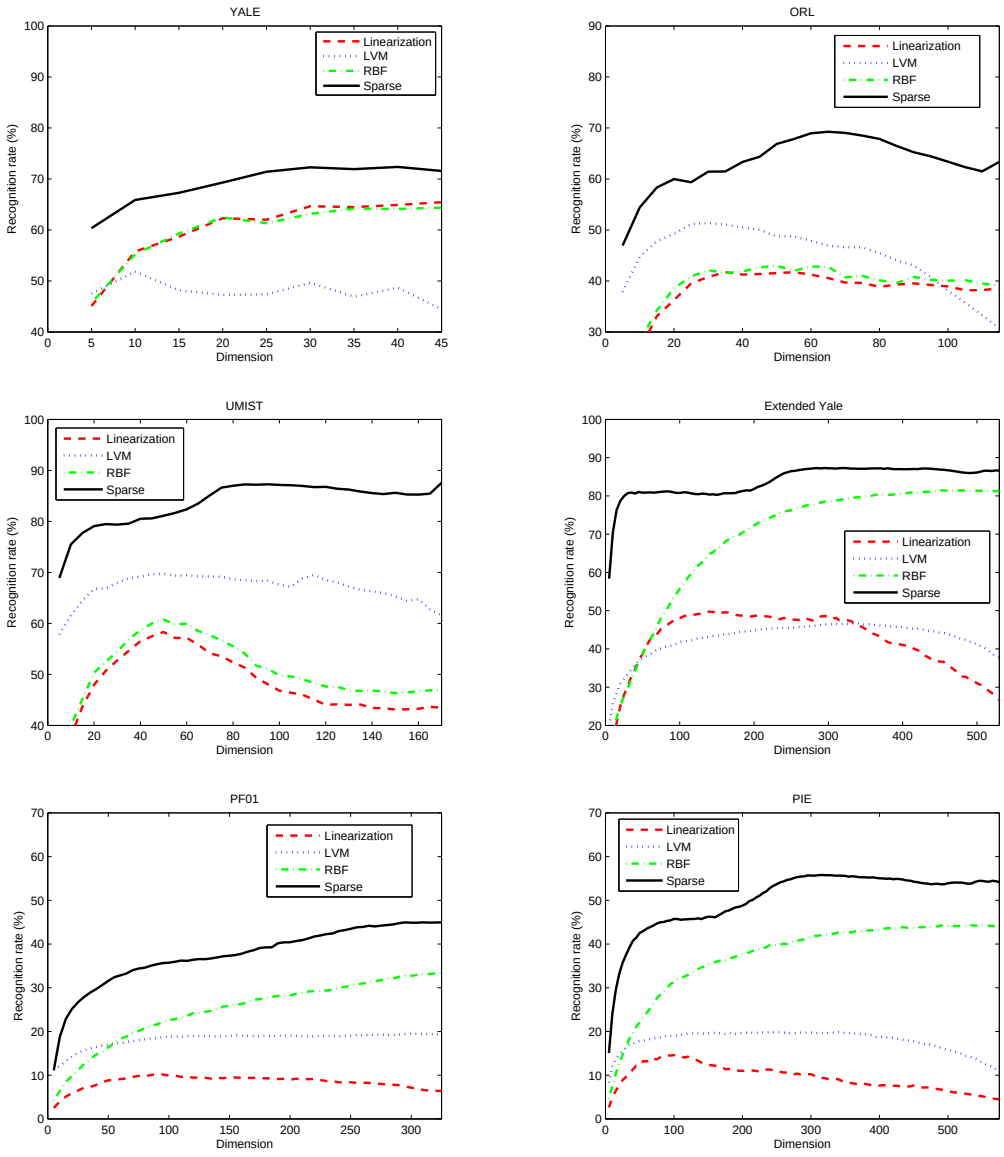

Figure 8: Experimental results on all 6 datasets for the 30-70 modality. The used classifier was 1 NN.

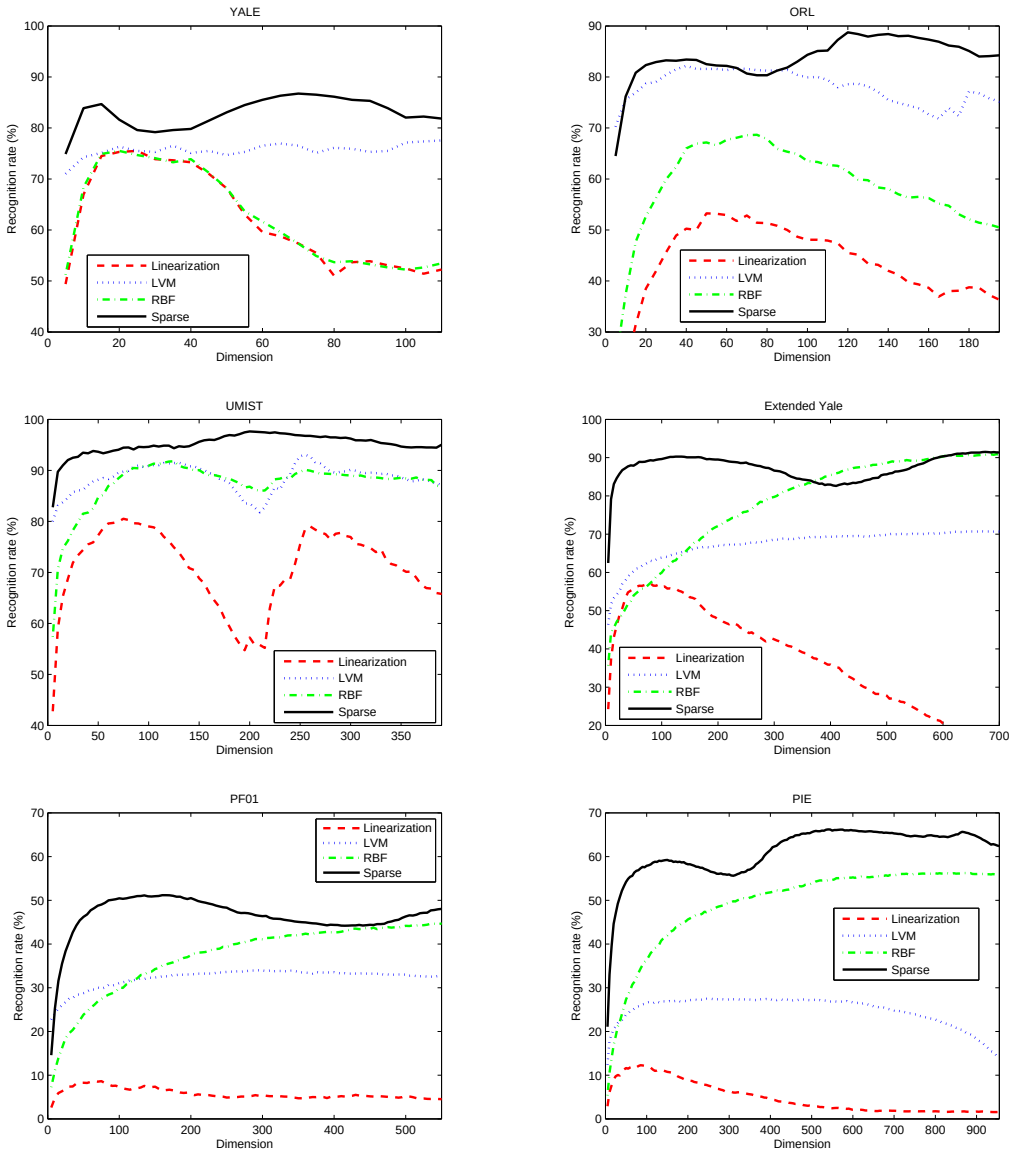

Figure 9: Experimental results on all 6 datasets for the 70-30 modality. The used classifier was 1 NN.

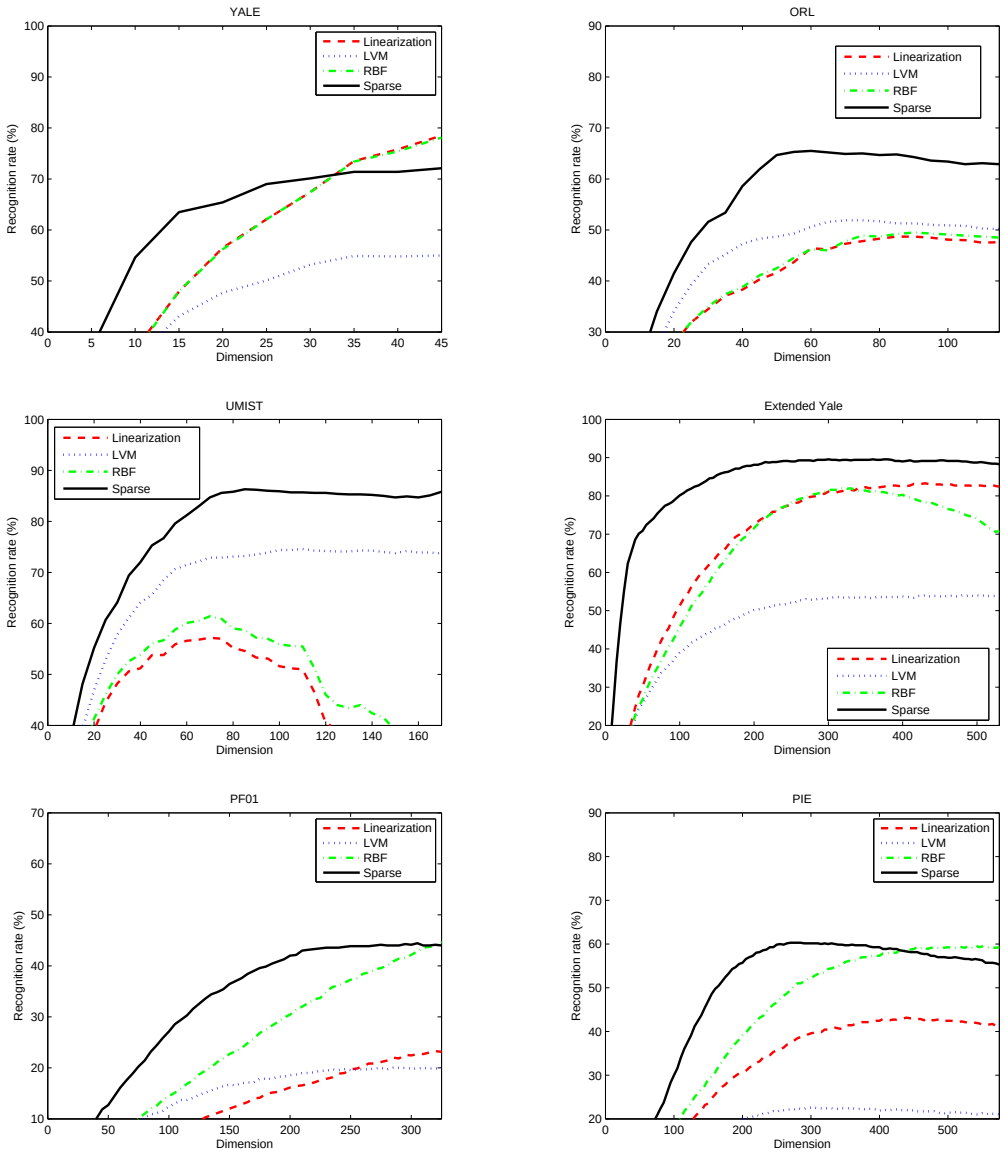

Figure 10: Experimental results on all 6 datasets for the 30-70 modality. The used classifier was SVM.

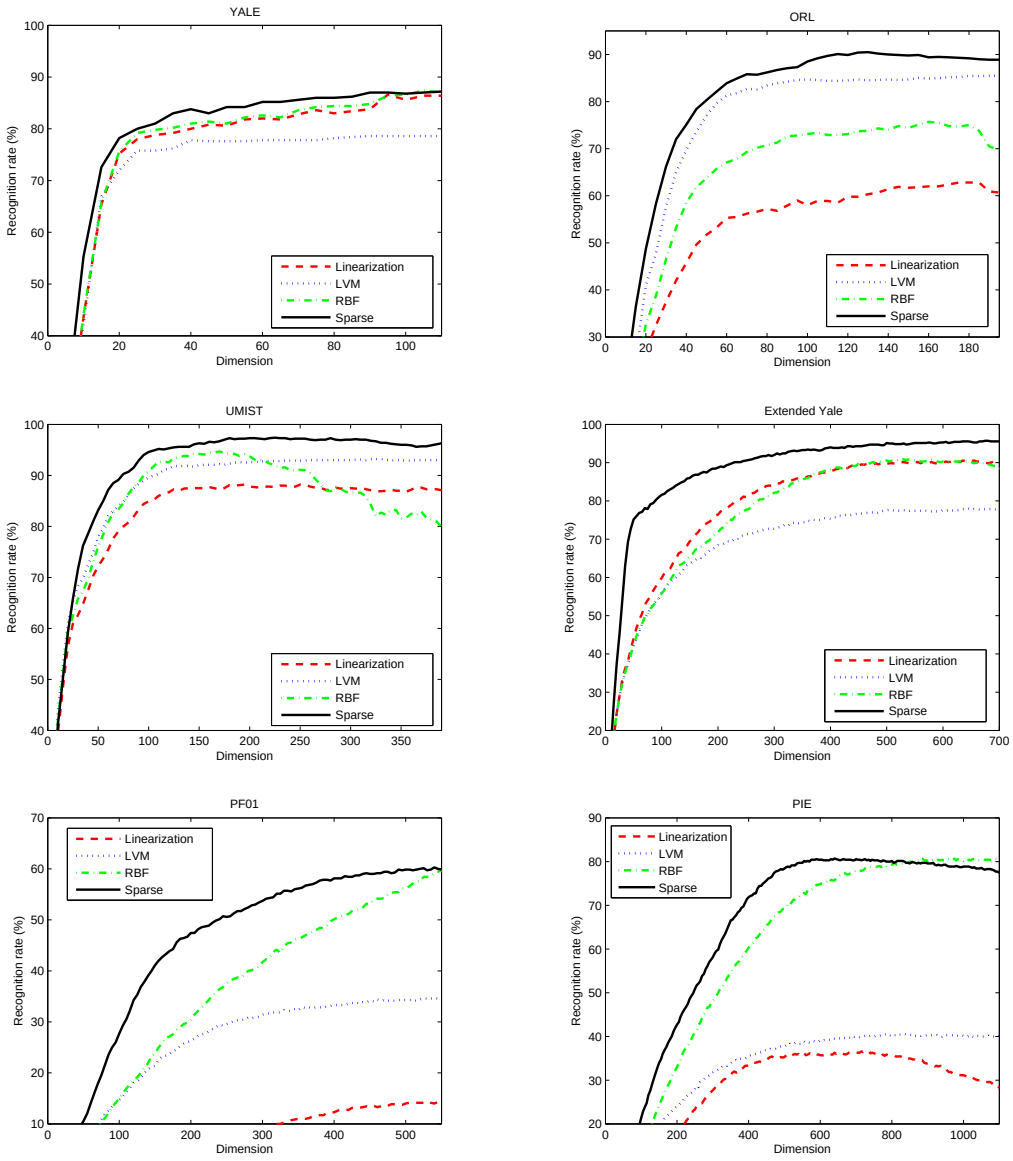

Figure 11: Experimental results on all 6 datasets for the 70-30 modality. The used classifier was SVM.

Dataset \ Method	Sparse Rep.	LVM			Lineariz.	RBF
30%-70%		$\epsilon = 3$	$\epsilon = 5$	$\epsilon = 7$		
YALE	**72.36%**	51.84%	41.66%	33.15%	65.43%	64.38%
ORL	**69.25%**	51.35%	37.71%	30.25%	41.71%	43.00%
UMIST	**87.56%**	69.72%	60.49%	52.65%	58.31%	60.76%
Ext. Yale	**87.29%**	46.66%	31.33%	24.25%	49.90%	81.45%
PF01	**45.00%**	19.50%	13.41%	10.36%	10.22%	34.58%
PIE	**55.79%**	19.86%	13.75%	11.18%	14.58%	44.35%
50%-50%		$\epsilon = 3$	$\epsilon = 5$	$\epsilon = 7$		
YALE	**81.85%**	70.12%	61.60%	52.09%	68.14%	67.90%
ORL	**82.50%**	72.05%	60.35%	49.25%	46.38%	52.44%
UMIST	**95.03%**	85.90%	76.04%	70.03%	76.25%	83.61%
Ext. Yale	**91.46%**	61.09%	46.85%	39.03%	53.14%	89.61%
PF01	**52.65%**	27.32%	20.40%	20.23%	8.27%	42.10%
PIE	**66.20%**	27.47%	20.57%	16.79%	12.26%	56.24%
70%-30%		$\epsilon = 3$	$\epsilon = 5$	$\epsilon = 7$		
YALE	**86.73%**	77.15%	73.87%	67.95%	75.51%	75.51%
ORL	**88.75%**	82.16%	73.66%	65.41%	53.25%	68.66%
UMIST	**97.74%**	93.06%	85.20%	79.94%	80.52%	91.79|%
Ext. Yale	92.12%	70.97%	58.36%	48.74%	57.14%	**92.49%**
PF01	**54.41%**	33.99%	27.06%	21.52%	8.66%	47.85%
PIE	**72.82%**	35.39%	26.78%	21.83%	13.42%	64.03%

Table 1: Maximum average recognition rate using the Nearest Neighbor classifier.

Dataset \ Method	Sparse Rep.	LVM			Lineariz.	RBF
30%-70%		$\epsilon = 3$	$\epsilon = 5$	$\epsilon = 7$		
YALE	72.10%	55.00%	44.00%	36.30%	**78.50%**	78.10%
ORL	**65.50%**	51.90%	40.80%	31.50%	48.70%	49.50%
UMIST	**86.30%**	74.60%	63.10%	55.00%	57.20%	61.40%
Ext. Yale	**89.57%**	54.00%	45.57%	37.57%	83.28%	82.14%
PF01	44.42%	20.00%	14.71%	10.42%	24.57%	**50.28%**
PIE	**60.28%**	22.57%	17.71%	12.85%	43.14%	59.71%
50%-50%		$\epsilon = 3$	$\epsilon = 5$	$\epsilon = 7$		
YALE	81.00%	71.70%	65.80%	56.20%	87.20%	**87.40%**
ORL	**82.60%**	75.40%	64.00%	52.80%	53.70%	62.60%
UMIST	**94.40%**	89.00%	80.80%	72.90%	79.50%	86.90%
Ext. Yale	**94.14%**	68.85%	63.00%	56.42%	88.28%	88.00%
PF01	53.42%	28.71%	23.00%	17.42%	20.14%	**62.28%**
PIE	73.28%	31.57%	23.57%	18.00%	38.57%	**74.28%**
70%-30%		$\epsilon = 3$	$\epsilon = 5$	$\epsilon = 7$		
YALE	87.20%	78.60%	78.00%	72.80%	86.60%	**87.40%**
ORL	**90.50%**	85.60%	77.00%	68.00%	62.80%	75.70%
UMIST	**97.40%**	93.20%	88.00%	81.70%	88.20%	94.70%
Ext. Yale	**95.71%**	78.57%	75.00%	69.57%	90.57%	91.00%
PF01	60.28%	35.85%	29.14%	22.71%	15.71%	**72.14%**
PIE	**80.71%**	40.57%	30.14%	23.71%	36.71%	**80.71%**

Table 2: Maximum average recognition rate using the SVM classifier.

	Sparse Rep.	LVM	Linearization	RBF
30%-70%				
YALE	**0.4956**	0.5300	0.5322	0.5306
ORL	**0.4915**	0.5227	0.6343	0.6319
UMIST	**0.4337**	0.5069	0.7181	0.7016
Ext. Yale	**0.5086**	0.6470	0.7699	0.6384
PF01	**0.5833**	0.6564	0.8095	0.6466
PIE	**0.6981**	0.7580	0.8047	0.6352
50%-50%				
YALE	**0.3786**	0.4429	0.4578	0.4549
ORL	**0.3597**	0.4058	0.9614	0.7167
UMIST	**0.3470**	0.4103	0.5834	0.5564
Ext. Yale	**0.3564**	0.3892	0.7528	0.4958
PF01	**0.4040**	0.4746	0.7746	0.4433
PIE	**0.4248**	0.4514	0.7758	0.4462
70%-30%				
YALE	**0.2685**	0.3173	0.3540	0.3486
ORL	**0.2520**	0.2686	0.3623	0.3440
UMIST	**0.2442**	0.2675	0.3722	0.3296
Ext. Yale	**0.1982**	0.2174	0.6658	0.3534
PF01	**0.2474**	0.2647	0.6707	0.2639
PIE	**0.2474**	0.2607	0.6794	0.2587

Table 3: Alignment error between batch-mode manifold and the out-of-sample computed manifold (See text for details.)

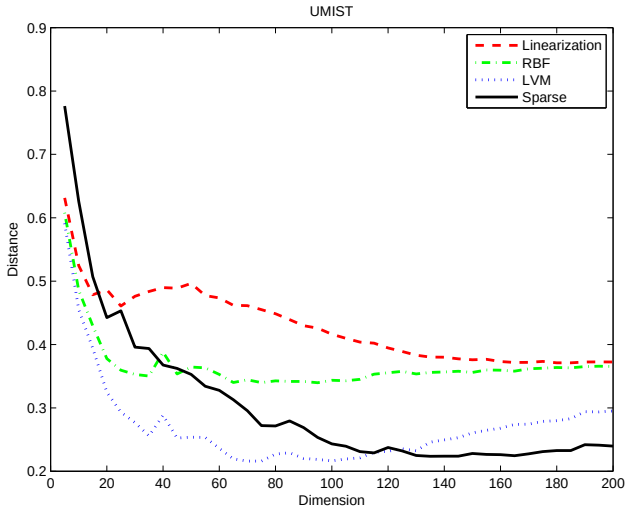

Figure 12: Variation of the alignment error as a function of the embedded space dimensionality

Method	*Sparse Rep.*	LVM	Linearization	RBF
CPU time	840.52	6.07	5.50	139.42

Table 4: CPU times (in milliseconds) representing the projection of one sample on the embedded space. The dimension of the input data is (D = 200), the dimension of the non-linear space is (d=1200), the number of training samples is (N=1241).

D	80	100	120	140	160	180	200
CPU Time	609.75	647.27	677.30	724.20	772.98	804.87	840.52

Table 5: CPU times (in milliseconds) representing the projection of a new sample on the embedded space (dimensionality d= 1200), as a function of the dimension of input data, D (using the Sparse Representation method).

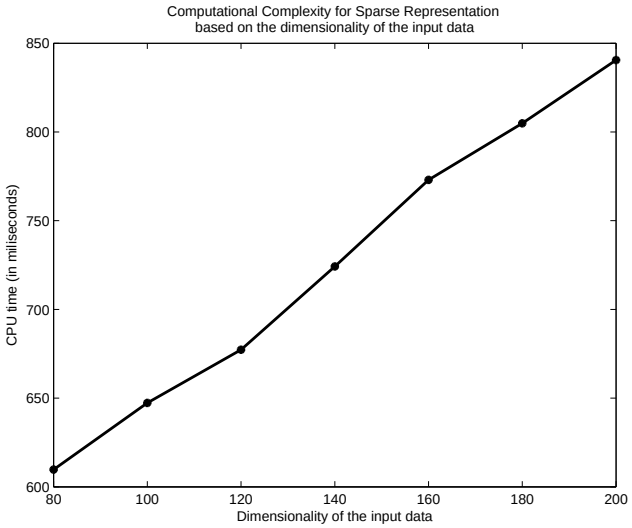

Figure 13: Variation of the CPU time (in milliseconds) of the Sparse Representation method as a function of the dimensionality of the input data. The CPU time corresponds to the projection of a new sample on the embedded space.

N	600	700	800	900	1000	1100	1200
CPU Time	169.68	219.23	285.85	351.48	447.37	513.04	647.27

Table 6: CPU times (in milliseconds) representing the projection of a new sample on the embedded space, as a function of the size of the training data, N (using the Sparse Representation method). The dimension of input data is kept fixed to D=100.

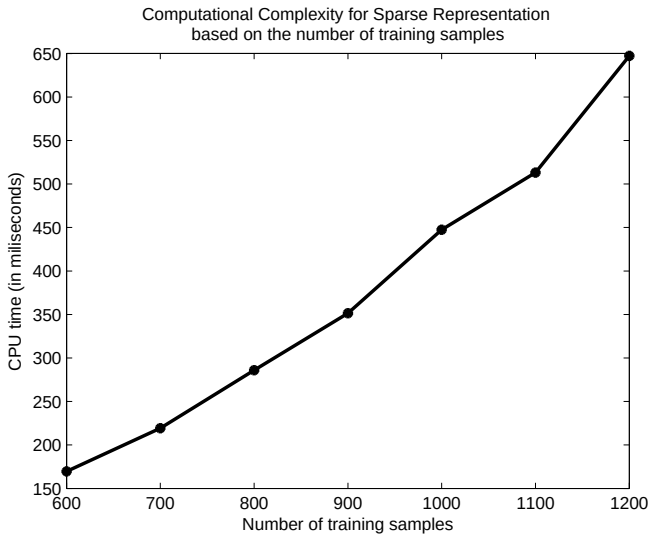

Figure 14: Variation of the CPU time (in milliseconds) of the Sparse Representation method as a function of the size of the training data. The CPU time corresponds to the projection of a new sample on the embedded space.

References

[1] B. Schölkopf, A. Smola, K.-R. Müller, Nonlinear component analysis as a kernel eigenvalue problem, Neural Computation 10 (1998) 1299–1319.

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

[3] L. K. Saul, S. T. Roweis, Y. Singer, Think globally, fit locally: Unsupervised learning of low dimensional manifolds, Journal of Machine Learning Research 4 (2003) 119–155.

[4] J. B. Tenenbaum, V. de Silva, J. C. Langford, A global geometric framework for nonlinear dimensionality reduction, Science 290 (5500) (2000) 2319–2323.

[5] X. Geng, D. Zhan, Z. Zhou, Supervised nonlinear dimensionality reduction for visualization and classification, IEEE Transactions on systems, man, and cybernetics-part B: cybernetics 35 (2005) 1098–1107.

[6] M. Belkin, P. Niyogi, Laplacian eigenmaps for dimensionality reduction and data representation, Neural Computation 15 (6) (2003) 1373–1396.

[7] P. Jia, J. Yin, X. Huang, D. Hu., Incremental Laplacian Eigenmaps by preserving adjacent information between data points, Pattern Recognition Letters 30 (16) (2009) 1457–1463.

[8] D. Donoho, C. Grimes, Hessian eigenmaps: Locally linear embedding techniques for high-dimensional data, in: Proc. of the National Academy of Arts and Sciences, 2003.

[9] I. Borg, P. Groenen, Modern Multidimensional Scaling: theory and applications, Springer-Verlag New York, 2005.

[10] K. Q. Weinberger, L. K. Saul, Unsupervised learning of image manifolds by semidefinite programming, International Journal of Computer Vision 70 (1) (2006) 77–90.

[11] S. Yan, D. Xu, B. Zhang, H. Zhang, Q. Yang, S. Lin, Graph embedding and extension: a general framework for dimensionality reduction, IEEE Trans. on Pattern Analysis and Machine Intelligence 29 (1) (2007) 40–51.

[12] S. Yan, H. Wang, Semi-supervised learning by sparse representation, in: SIAM International Conference on Data Mining, 2009.

[13] M. W. Trosset, C. E. Priebe, The out-of-sample problem for classical multidimensional scaling, Comput. Stat. Data Anal. 52 (10) (2008) 4635–4642.

[14] B. Raducanu, F. Dornaika, A supervised non-linear dimensionality reduction approach for manifold learning, Pattern Recognition 45 (2012) 2432–2444.

[15] A. Elgammal, C. Lee, Non-linear manifold learning for dynamic shape and dynamic appearance, Computer Vision and Image Understanding 106 (1) (2007) 31–46.

[16] Y. Yang, F. Nie, S. Xiang, Y. Zhuang, W. Wang, Local and global regressive mapping for manifold learning with out-of-sample extrapolation, in: American Association for Artificial Intelligence Conference, 2010.

[17] C. Piret, Analytical and numerical advances in radial basis functions, Ph.D. thesis, University of Colorado at Boulder (2007).

[18] M. Scheuerer, An alternative procedure for selecting a good value for the parameter c in rbf-interpolation, Advances in Computational Mathematics 34 (1) (2011) 105–126.

[19] P. Dollar, V. Rabaud, S. Belongie, Learning to traverse image manifolds, in: NIPS, 2006.

[20] Y. Bengio, J. Paiement, P. Vincent, Out-of-sample extensions for LLE, Isomap, MDS, eigenmaps and spectral clustering, in: Advances in Neural Information Processing, 2004.

[21] J. Verbeek, Learning nonlinear image manifolds by global alignment of local linear models, in: IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 28, 2006, pp. 1236–1250.

[22] T.-J. Chin, D. Suter, Out-of-sample extrapolation of learned manifolds, IEEE Transactions on Pattern Analysis and Machine Intelligence 30 (2008) 1547–1556.

[23] H. Strange, R. Zwiggelaar, A generalised solution to the out-of-sample extension problem in manifold learning, in: American Association for Artificial Intelligence Conference, 2011.

[24] L. Zhan, L. Qiao, S. Chen, Graph-optimized locality preserving projections, Pattern Recognition 43 (2010) 1993–2002.

[25] Y. Xu, A. Zhong, J. Yang, D. Zhang, LPP solution schemes for use with face recognition, Pattern Recognition 43 (2010) 4165–4176.

[26] J. Liu, Face recognition on Riemannian manifolds, Master's thesis, Autonomous University of Barcelona (2011).

[27] D. L. Donoho, M. Elad, V. Temlyakov, Stable recovery of sparse overcomplete representations in the presence of noise, IEEE Trans. Information Theory 52 (1) (2006) 6–18.

[28] J. Wright, A. Yang, A. Ganesh, S. Sastry, Y. Ma, Robust face recognition via sparse representation, IEEE Trans. on Pattern Analysis and Machine Intelligence 31 (2) (2009) 210–227.

[29] M. Elad, Sparse representations are most likely to be the sparsest possible, Journal on Applied Signal Processing 2006 (2006) 1–12.

[30] J.-B. Huang, M.-H. Yang, Fast sparse representation with prototypes, in: IEEE Conference on Computer Vision and Pattern Recognition, 2010, pp. 3618–3625.

[31] J. Shi, J. Malik, Normalized cuts and image segmentation, IEEE Trans. on Pattern Analysis and Machine Intelligence 22 (8) (2000) 888–905.

[32] A. Ng, M. Jordan, Y. Weiss, On spectral clustering: Analysis and an algorithm, in: NIPS 14, 2002.

[33] S. X. Yu, J. Shi, Multiclass spectral clustering, in: IEEE International Conference on Computer Vision, 2003.

[34] M. A. Carreira-Perpinan, Z. Lu, The Laplacian Eigenmaps latent variable model, Journal of Machine Learning Research 2 (2007) 59–66.

[35] M. Zibulevsky, M. Elad, L1-L2 optimization in signal and image processing, IEEE Signal Processing Magazine 27 (3) (2010) 76–88.

[36] E. Candes, J. Romberg, l1-magic: recovery of sparse signals via convex programming, CALTECH, http://www.acm.caltech.edu/l1magic/ (2005).

[37] N. Goel, G. Bebis, A. Nefian, Face recognition experiments with random projections, in: SPIE Conference on Biometric Technology for Human Identification, 2005.

[38] D. Cai, X. He, J. Han, Spectral regression for efficient regularized subspace learning, in: Proc. Int. Conf. Computer Vision (ICCV'07), 2007.

[39] G. Golub, C. Van Loan, Matrix Computations, Johns Hopkins University Press, Baltimore, MD, 1996.