Locally Linear Embedding and its Variants for Feature Extraction

Róbert Busa-Fekete, András Kocsor[*]
Research Group on Artificial Intelligence of the Hungarian Academy of Sciences and University of Szeged
H-6720 Szeged, Aradi vértanúk tere 1., Hungary
{busarobi,kocsor}@inf.u-szeged.hu

Abstract ***– Many problems in machine learning are hard to manage without applying some pre-processing or feature extraction method. Two popular forms of dimensionality reduction are the methods of principal component analysis (PCA) [2] and multidimensional scaling (MDS) [18]. In this paper we examine Locally Linear Embedding (LLE), which is an unsupervised, non-linear dimension reduction method that was originally proposed for visualisation. We will show that LLE is capable of feature extraction if we choose the right parameter values. In addition, we extend the original algorithm for more efficient classification. Afterwards we apply the methods to several databases that are available at the UCI repository, and then show that there is a significant improvement in classification performance.***

Keywords ***– Locally Linear Embedding, feature extraction methods, non-linear feature map, parameter selection algorithms***

I. INTRODUCTION

Classification algorithms require that the objects to be classified should be represented as points in a multidimensional feature space. However, before executing a learning algorithm, additional vector space transformations may need to be applied on the initial feature data. The reason for doing this is twofold: firstly these transformations can improve classification performance and, secondly, they can reduce the dimensionality of the data (i.e. we can avoid the so-called 'curse of dimensionality' problem). In the literature sometimes both the choice of the initial features and their transformations are dealt with under the heading "feature extraction". To avoid any misunderstanding here we will cover only the latter, namely the transformation of the initial feature set into another one. This, it is hoped, will yield a more efficient or, at least, faster classification procedure.

In this paper we consider the Locally Linear Embedding (LLE) method, which was first introduced by Sam T. Roweis and Lawrence K. Saul in 2000 [1]. It is an unsupervised, non-linear dimensionality reduction procedure. The aim here is to show how we can apply LLE as a feature extraction technique if we choose suitable parameter values.

We will also introduce and describe here some novel and efficient versions of the original procedure. First of all we combine the traditional Principal Component Analysis (PCA) procedure [2] with LLE, which results in a method that is less sensitive to linearly dependent input vectors. We will also introduce a supervised version of the original unsupervised LLE method by redefining the distances between data points according to their class labels. Since it is necessary to express the distances and the covariance matrix of LLE in a kernel feature space (see Section III) we will also introduce the kernel version of the method. Along the way by modifying the kernel functions we can get alternative methods. After applying the proposed methods on the same data sets of various sizes and making comparisons, we find that they are generally beneficial prior to classification.

The structure of the paper is as follows. The original method [1] will be described briefly in Section II. Then we will present the above versions of the original LLE method in Section III. In Section IV we will discuss the issue of parameter selection for the proposed algorithms, and we then round off the paper with some concluding remarks in Section V.

II. THE ORIGINAL LOCALLY LINEAR EMBEDDING

In this section we will briefly describe the original LLE method. While doing this we can also get an insight into the mathematical background of the original algorithm and the role of the parameters.

A. The LLE method

The main idea behind LLE is to embed the objects, which are represented as multidimensional points, into a lower dimensional space while preserving their neighbourhoods. With this approach the neighbourhoods will be retained in the following way: in the first step every data point is expressed as a linear combination of its K-nearest neighbours and, in the second step, the image points in the embedded space are reconstructed using the weights

[*] The author was supported by the János Bolyai fellowship of the Hungarian Academy of Sciences.

of the linear combination that were determined in the higher dimensional space.

Before we discuss the LLE algorithm in detail, let us represent the input data as D-dimensional points in a Euclidean vector space: $x_1, \ldots, x_n \in \mathbb{R}^D$, and represent the output vectors as $y_1, \ldots, y_n \in \mathbb{R}^d$, where $d \ll D$

First step: As we mentioned above we need to express every point as a linear combination of its neighbours. Now let $w_{ij} \in \mathbb{R}^n$ denote the *j*th weight of the linear combination for the *i*th point. To solve the above problem, we need to minimise the following expression:

$$\Phi(W) = \sum_{i=1}^{n} \left\| x_i - \sum_{j=1}^{n} w_{ij} x_j \right\|_2^2, \qquad (1)$$

where we have represented the weights in matrix form (the *i*th row corresponds to the w_i weight vector for the *i*th point). Because we only consider the *K*-nearest neighbours in the Euclidean sense for every point, every weight vector w_i can contain at most *K* non-zero elements. This means that the matrix *W* will be mostly sparse. Afterwards, without loss of generality, we may assume that the weights sum up to one for each point, which will make the optimisation problem easier.

Second step: In this part of the procedure we would like to determine the image points in an optimal way as they reflect the structure of the input dataset. Hence it seems obvious that we should also minimise a similar expression for the image points, but with the unknown matrix *Y* containing the output vectors as its columns:

$$\Psi(Y) = \sum_{i=1}^{n} \left\| y_i - \sum_{j=1}^{n} w_{ij} y_j \right\|_2^2 \qquad (2)$$

The minimum value here is invariant under the rotations and translations of the image points. So let us assume that the image points have been centralised, i.e. $\sum_{i=n}^{n} y_i = 0$ and have unit covariance, i.e. $\frac{1}{n}\sum_{i=1}^{n} y_i y_i^T = I$, where *I* is the identity matrix of size *dxd*. Solving this minimisation problem, we can then determine the embedded image points.

A good example of embedding a large three-dimensional dataset into two dimensions is shown below, which was first presented by the authors of the LLE method [19]. This 3-dimensional dataset gives us an indication of how the points actually remain in their neighbourhoods.

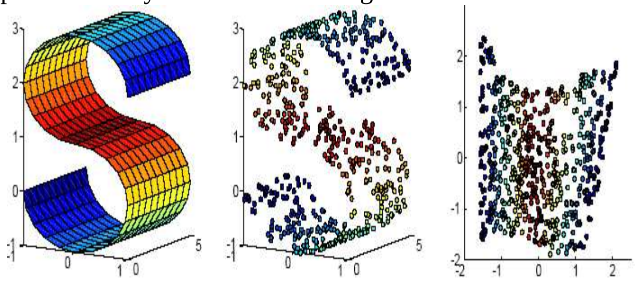

Fig. 1. LLE maps a 3-dimensional dataset into a 2-dimensional one

B. Solving the two LLE optimisation problems

Both steps lead us to an optimisation problem that is called the constrained least squares problem. Thus we really need only solve the same general problem.

First step. We need to minimise the expression in Eq. (1) subject to the constraint that the weights vectors w_i sum up to one. We can solve this equation for each point separately. Let $N(1), \ldots, N(K)$ denote the indices of the *K*-nearest neighbours for a fixed point, then we can reexpress Eq. (1) in the following form:

$$\left\| \mathrm{x_i} - \sum_{j=1}^{K} w_{iN(j)} x_{N(j)} \right\|_2^2 = \left\| \sum_{j=1}^{K} w_{iN(j)} (x_i - x_{N(j)}) \right\|_2^2 \qquad (3)$$

$$= \sum_{j=1}^{K} \sum_{k=1}^{K} w_{iN(j)} w_{iN(k)} c_{jk}^i, \qquad (4)$$

where we exploited the fact that the weights must sum up to one, and then we rewrote the second term as the local covariance matrix

$$c_{jk}^i = (x_i - x_{N(j)})^T \cdot (x_i - x_{N(k)}) \qquad (5)$$

Now that this expression has a closed form it is fairly straightforward to find the minimum value. A good way of doing this is by looking for the stationary point of the Lagrange function:

$$L_1(w, \lambda) = \sum_{j=1}^{K} \sum_{k=1}^{K} w_{iN(j)} w_{iN(k)} c_{jk} + \lambda\left(\sum_{j=1}^{K} w_{iN(j)} - 1\right), \qquad (6)$$

where λ is the Lagrange multiplier. In practice if we solve the linear equation $Cv_i = 1$ for every point where v_i corresponds to the *K* unknown elements in the vector w_i, then we will obtain the same result, and it is easier to compute. We should mention, however, that the C matrix might sometimes be singular. To overcome this problem we can add a small term to the C matrix:

$$C := C + rI, \qquad (7)$$

where *r* is a small regularisation parameter that will have only a negligible effect on the results.

Second step: We may rewrite the target function represented in Eq (2). If we introduce the matrix M which has the following form

$$M = (I - W)^T (I - W), \qquad (8)$$

then the problem can be written in a more compact form. That is,

$$\Psi(Y) = tr(YMY^T), \qquad (9)$$

where the matrix *Y* contains the image points as its columns. And like the first problem, it can also be solved using the Lagrange Multiplier Method, where the Lagrange function now has the form

$$L_2(Y, \lambda) = tr(YMY^T) + tr(\Lambda YY^T - nI) \qquad (10)$$

Here the matrix Λ contains the Lagrange multipliers as its diagonal elements. It is not hard to see that the stationary points of this expression are the eigenvectors of the M matrix, and its values are the corresponding eigenvalues.

One of the above-mentioned constraints was not included in the optimisation problem because if we also have the constraint $\sum_{i=n}^{n} y_i = 0$ we can discard the eigenvector with the smallest eigenvalue. In short, we can get the desired optimal image points if we write the *d+1* eigenvectors of the matrix *M* in the matrix *Y* as its rows and leave out the eigenvector with the smallest eigenvalue.

C. Choosing the parameters

When using LLE we have to choose values for three parameters. These are the number of neighbours considered (*K*), the dimension of the image space (*d*) and the regularisation parameter (*r*).

Perhaps the most important parameter is the image dimension of space, because it can have a big influence on the classification accuracy. Here we recommend a heuristic for choosing a 'good' value. In the case of the PCA method the dimension of the image space is determined by an eigenanalysis of the *C* correlation matrix of the sample as follows. Let $\lambda_1, \ldots, \lambda_D$ stand for the eigenvalues of the matrix C. We then have to determine the dimension *d* of the image space, which must satisfy the following inequality:

$$v \le \frac{\sum_{i=1}^{d} \lambda_i}{\sum_{j=1}^{D} \lambda_j} \qquad (11)$$

For the PCA method it means that the ratio of the residual covariance and the original covariance of the sample should be higher than a given value *v*. In practice this value is usually fixed around *0.95-0.97.*

Using the above approach we could also apply this procedure to LLE. We could perform a similar eigenanalysis for all of the local covariance matrices, and then determine the dimension of image space using the inequality in Eq. (11), which must hold true for all points in the sample. If we do this the dimension of image space will not be too low. And since this procedure is automatic the LLE method can be applied to a wider range of problems.

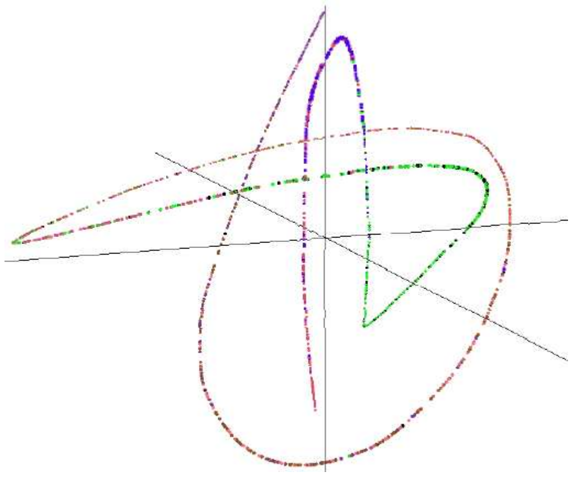
Fig. 2. The mfeat database from the UCI repository is mapped into 2-dimensions with a proper parameter selection (K=30, D=6, d=3, n=1800)

The number of neighbours considered is another parameter. If we set it too high, the LLE method will be slowed down. But in other circumstances it is worth choosing a *K* value larger than the dimension of the input space *D*, because it satisfies the condition for the local covariance matrix for a given point, namely $rank(C) = \min(D, K)$. Moreover, regularisation does not play such an important role in determining the weights in the first step described above. Figures 2 and 3 both show what will happen when we set the *K* value too low. The structure of the dataset breaks up, and the reconstruction error is higher than normal.

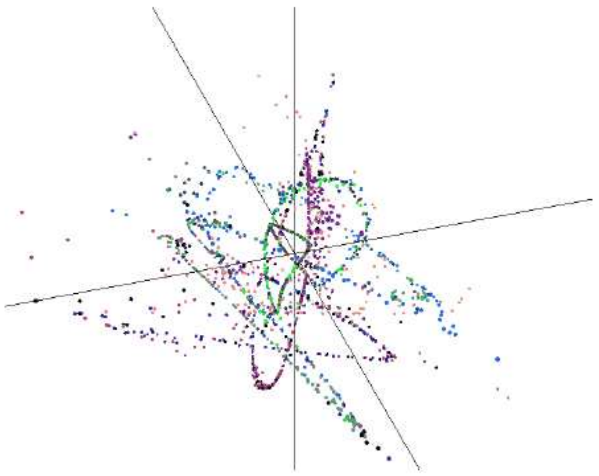
Fig. 3. The mfeat database from the UCI repository is mapped into 2-dimensions with a poor choice of K (K=8, D=6, d=3, n=1800)

The last parameter that can be varied is the regularisation parameter. The regularisation trick guarantees that the local covariance matrices will not be non-singular in the first step of LLE. Regularisation translates the eigenvalues by *r*, so we should make the value of *r* smaller than the smallest eigenvalue of the local covariance matrices.

III. EXTENSIONS OF LLE, AND THEIR APPLICATIONS TO FEATURE EXTRACTION

In this section we introduce some extensions of the original LLE approach. Afterwards, we will say how they can be applied to feature extraction.

A. The combination of PCA and LLE (PCA-LLE)

The first supervised extension of LLE that we will describe here is a combination of PCA and the original LLE method. Instead of using the *K*-nearest neighbours, we may reconstruct the points with specific directions that will be determined by the PCA method. We may obtain a better reconstruction, so we can determine the image points using more precise weights.

Before giving a formal description of PCA-LLE, let us assume as well that we have k classes and an indicator function:

$$\tau : \{1, \ldots, n\} \to \{1, \ldots, k\},$$

where $\tau(i)$ gives the class label of the sample x_i. Furthermore, let X_i denote the matrix that contains the elements of the sample with class label i, and C_i its correlation matrix, assuming that the probabilities of the elements are equal in the *i*th class. We will use this notation later in this paper when we talk about the supervised case.

Next we should calculate the directions that the PCA method determines for a given class i, the so-called principal components. It means that we have determine the eigenvectors of the correlation matrix C_i of size *DxD* for any class. Then we use these directions in Eq. (3) instead of the K-nearest neighbours. The other steps of LLE remain unchanged.

It may sometimes be a problem if the dataset has a very high dimension because the eigenanalysis of the correlation matrices may require much more computation. But we do not need to use all of the principal components, because we could consider only the *K* biggest one, and then PCA-LLE will work well. This extra-computation is, however, negligible for the eigenanalysis in the second step of LLE.

B. Modified Supervised Locally Linear Embedding (MSLLE)

The second supervised extension that will be examined here is the MSLLE method. It is based on the idea that we should make the class more separable before we apply LLE on the dataset. It can be managed with a redefinition of the distance matrix in such a way that the elements in the different classes have distances that are inversely proportion to each other.

First of all we need to calculate the local covariance matrix C using the distances between the elements. Let d_{ij} stand for the distance between the elements i and j, and let $D_{\max} = \max_{1\le i,j\le n}\left(d_{ij}\right)$. Now let us introduce a new spacing parameter α. This allows us to redefine the distances in the following way:

$$d_{ij} := \begin{cases} d_{ij} & ,if\, \tau(i)=\tau(j) \\ d_{ij}+\alpha(D_{\max}-d_{ij}) & ,otherwise \end{cases} \qquad (12)$$

It is easy to see that two elements that belong to different classes will have bigger distances, while those in the same class will not. But the distances of the former will be altered by only a small amount if they are relatively far from each other. Using a physics metaphor, points in different classes are repelled with a force law that is inversely proportional to the distance between them, but points in the same class are unaffected.

In this case it is more convenient if we calculate the local covariance matrix in the following form:

$$c^i_{jk} = \frac{1}{2}\left(d_{i,N(j)} + d_{i,N(k)} + d_{N(j)N(k)}\right). \qquad (13)$$

We get a new method called MSLLE where we can increase the separation between the classes in the embedded space. We propose this method only for feature extraction, because this method does not preserve the neighbourhoods between the elements. So it is not a suitable procedure for visualisation.

C. Kernel Locally Linear Embedding (KLLE)

The kernel idea can be applied in cases where the input of some algorithm consists of the pairwise dot (scalar) products of the elements of an n-dimensional dot product space. In this case, simply by a proper redefinition of the two-operand operation of the dot product, we can have an algorithm that will now be executed in a different dot product space, which is probably more suitable for solving the original problem. Of course, when replacing the operand, we have to satisfy certain criteria, because not every function is suitable for implicitly generating a dot product space. According to Mercer's theorem the family of the Mercer Kernels is an appropriate choice [20],[11].

The KLLE method is a kernelised version of the original LLE, which was introduced in [7]. Now let the dot product be implicitly defined by the kernel function κ in some finite or infinite dimensional feature space $\mathbb{F}$ with associated transformation ϕ:

$$\kappa(x,y) = \phi(x)\cdot\phi(y) \qquad (14)$$

Than we can express the local covariance matrix for a given point x_i:

$$c^{i,\phi}_{jk} = \kappa(x_i,x_i) - \kappa(x_i,x_{N(j)}) - \kappa(x_i,x_{N(k)}) + \kappa(x_{N(j)},x_{N(k)})$$

Using different kernel functions we get alternative algorithms in this way. For instance we could choose one of the two well-known kernel functions:

1. Polynomial kernel function:
$$\kappa_1(x,y) = (x^T y + \sigma)^q, q\in\mathbb{N}, \sigma\in\mathbb{R}^+$$
2. Rational quadratic kernel:
$$\kappa(x,y) = 1 - \frac{\|x-y\|^2}{\|x-y\|^2+\sigma}, \sigma\in\mathbb{R}^+$$

D. The methods as a feature extraction algorithms

In many areas of science LLE is very helpful (e.g. in data mining, bioinformatics), because it gives an insight into the structure of the dataset being examined. This method was originally intended for visualising the data. If we would like to use the original LLE for feature extraction, then we need to embed a new element into the image space. In the case of LLE there is no transformation that has a compact form. So we suggest the following straightforward method for determining the image of a new point *z*: let us find the *K*-nearest neighbours of *z* in the original space, and then let us calculate the weights according to the first step of the LLE method, and then determine the image of *z* as the linear combination of the image of the *K*-nearest neighbours of point *z*. All of the algorithms that are introduced in Section III can use this method for determine the image point of each new element.

IV. EXPERIMENTS

Our proposed methods were tested on many databases that are available at the UCI Repository [15]. We compared the performance of the algorithms with the help of three well-known classifiers [21]: Distance Weighted k-Nearest Neighbours (DKNN), Inverse Distance Weighted k-Nearest Neighbours (IDKNN) and Nearest Mean Classifier (NM). The accuracy of the classifiers was assessed using 10 fold cross validation.

A. Tests

The first dataset we examined is called wine, which contains 13 parameters about 178 different Italian wines. We set up the value of parameter $K = 20$, and the regularisation parameter $r = 10^{-5}$. The automatic method that we suggested in Section II for determining the dimension of the image space gives a result of $d = 10$ for all of the algorithms except KLLE. With the latter the feature space can be infinite, thus this automatic method will not work. So we also give d a value of 10 in KLLE. There is another parameter of the KLLE method: the kernel function. We used the polynomial kernel function with parameter with $q = 3, \sigma = 0.01$. In addition the α parameter was set to 0.3 in MSLLE. The results with these parameters can be seen in the table below.

Table I. The performance of the 3 distinct classifiers on the wine datasets using 10 fold cross validation

	DKNN	IDKNN	NM
Without feature extraction	93.33%	93.14%	**95.55%**
LLE	95.55%	**96.66%**	96.66%
MSLLE	**97.22%**	97.22%	97.22%
PCA-LLE	95.55%	**97.22%**	96.11%
KLLE	94.44%	95.00%	**96.66%**

The second database on which we tried the algorithms was called the heart_disease, which contains data about peoples who had heart disease. This database has the following parameters: $n = 294, K = 15$, and the embedding carried out had $d = 8, r = 10^{-4}$. Table II shows the results. The last row gives the classification performance using KLLE. After trying many kernels, the polynomial kernel seemed the best choice for this problem. The α parameter in the MSLLE method was set to 0.2.

Table II. The performance of the 3 distinct classifiers on the heart_disease datasets using 10 fold cross validation

	DKNN	IDKNN	NM
Without feature extraction	55.73%	**60.85%**	56.34%
LLE	**64.94%**	62.89%	62.48%
MSLLE	57.48%	57.75%	**60.62%**
PCA-LLE	**61.89%**	61.21%	60.17%
KLLE	**62.56%**	60.85%	61.87%

In Table III we can see the results of classification on the database internet_advertisements. This represents a set of possible advertisements on Internet pages. We chose this set because, the database contains 3279 instances, and the number of dimensions compared to the number of instances (D=1558) is relatively high, so it seemed ideal for using a feature extraction method. The embedding was carried out using the following parameters: $d = 50, K = 50, r = 10^{-3}$. We again used KLLE with a polynomial kernel function, with values $q = 3, \sigma = 0.01$, and MSLLE with a spacing value of $\alpha = 0.3$.

Table III. The performance of the 3 distinct classifiers on the internet_advertisements datasets using 10 fold cross validation

	DKNN	IDKNN	NM
Without feature extraction	88.23%	**89.85%**	88.07
LLE	90.88%	90.75%	**91.27%**
MSLLE	92.46%	92.68%	**92.81%**
PCA-LLE	91.01%	**91.27%**	91.27%
KLLE	**91.91%**	91.27%	90.32%

B. Evaluation

Using three different databases we showed how the algorithms work as a feature extraction method. In the first test (wine) it attained its highest value with MSLLE and PCA-LLE, but with PCA-LLE the choice of classifiers had a smaller effect on the performance. So the embedded dataset may be more reliable. In the second test (heart_disease) we can see that LLE achieved the highest score of all. Clearly this automatic method for determining the dimension of the embedded space also made the original LLE algorithm suitable for feature extraction. What is more, in this case the second best result was provided by KLLE. We should mention here that the choice of the kernel function significantly influences the accuracy, and it is no easy task to find a suitable one for a given classification problem. Finally, in the last test (internet_advertisements) the results again showed that the PCA-LLE method was the least sensitive to the choice of classifiers, but the best result was achieved by MSLLE. With the latter we had to choose an additional parameter α, and we found empirically that this algorithm gives a good performance when α has a value between 0.2-0.4.

V. CONCLUSIONS

The experiments demonstrated that the LLE method and its extensions are good not just for visualisation, but for feature extraction as well. We showed experimentally with datasets of various sizes that it is worth extending the original LLE method because we can achieve a significant improvement in performance for many classification problems.

In this paper we introduced a new combination of PCA and LLE, a supervised version of LLE and we reviewed a kernelised form of the original method [7]. We then carried out tests using three basic classifiers. These tests confirmed that it was indeed reasonable to extend the LLE method. In the future we intend to combine it with other feature extraction methods to learn more about the effect of combinations on the classification performance of datasets.

VI. REFERENCES

[1] L.K. Saul, S.T. Roweis: Nonlinear dimensionality reduction by locally linear embedding, Science, 290:2323-2326,2000.

[2] Jollife, I. T. (1986), Principal Component Analysis, Springer-Verlag

[3] M. A. Aizerman, E. M. Braverman, L. I. Rozonoer, "Theoretical foundation of the potential function method in pattern recognition learning," Automat. Remote Cont., Vol. 25, pp. 821-837, 1964.

[4] B. E. Boser, I. M. Guyon, V. N. Vapnik: A Training Algorithm for Optimal Margin Classifiers, in Proc. of the Fifth Annual ACM Conference on Computational Learning Theory, D. Haussler (eds.), ACM Press, Pittsburg, pp. 144-152, 1992.

[5] P.G. Ciarlet, J.L. Lious: Handbook of Numerical Analysis, North-Holland, Amsterdam.

[6] F. Cucker, S. Smale, "On the mathematical foundations of learning," Bull. Am. Math. Soc., Vol. 39, pp. 1-49, 2002.

[7] D. DeCoste : Visualizing Mercer Kernel Feature Spaces Via Kernelized Locally Linear Embeddings, Machine Learning System Group, Jet Propulsion Laboratory, California Institute of Technology, M/S 126/347, 4800 Oak Grove Drive, Pasadena, CA 91109, USA

[8] J. Ham, D.D. Lee, S. Mika, B. Schölkopf: Kernel view of the dimensionality reduction of manifolds, Technical Report No. TR-110, Max Planck Institute for Biological Cybernetics, 2003

[9] D. J. Hand, Kernel discriminant analysis, Research Studies Press, New York, 1982.

[10] Kernel Machines Web site, http://kernel-machines.org.

[11] J. Mercer: Functions of positive and negative type and their connection with the theory of integral equations, Philos. Trans. Roy. Soc. London, A, Vol. 209, pp. 415-446, 1909.

[12] S.N. Mukherjee: Locally Linear Embedding for Speech Recognition, Churchill College, University of Cambridge, 2002

[13] E. Parzen: On the estimation of the probability density function and mode, Annals of Mathematical Statistics, Vol. 33, pp. 1065-1076, 1962.

[14] http://www.cs.toronto.edu/~roweis/lle/code.html

[15] http://www.ics.uci.edu/ mlearn/MLRepository.html

[16] V. N. Vapnik, The Nature of Statistical Learning Theory, Springer, New York, 1995.

[17] V. N. Vapnik, Statistical Learning Theory, John Wiley & Sons Inc., 1998.

[18] T. Cox and M. Cox. Multidimensional Scaling (Chapman & Hall, London, 1994).

[19] L.K. Saul, & S.T. Roweis: Think Globally, Fit Locally Unsupervised Learning of Non-linear Manifolds, Technical Report MS CIS-02-18, University of Pennsylvania, 2002.

[20] F. Cucker, S. Smale: On the mathematical foundations of learning, Bull. Am. Math. Soc., Vol. 39, pp. 1-49, 2002.

[21] Richard O. Duda, Peter E. Hart and David G. Stork, Pattern Classification, Wiley & Sons (2001).