Incremental Locally Linear Embedding Algorithm

Olga Kouropteva*, Oleg Okun, and Matti Pietikäinen

Machine Vision Group,
Infotech Oulu and Department of Electrical and Information Engineering,
P.O.Box 4500, FI-90014 University of Oulu, Finland
{kouropte, oleg, mkp}@ee.oulu.fi

Abstract. A number of manifold learning algorithms have been recently proposed, including locally linear embedding (LLE). These algorithms not only merely reduce data dimensionality, but also attempt to discover a true low dimensional structure of the data. The common feature of the most of these algorithms is that they operate in a batch or offline mode. Hence, when new data arrive, one needs to rerun these algorithms with the old data augmented by the new data. A solution for this problem is to make a certain algorithm online or incremental so that sequentially coming data will not cause time consuming recalculations. In this paper, we propose an incremental version of LLE and experimentally demonstrate its advantages in terms of topology preservation. Also, compared to the original (batch) LLE, the incremental LLE needs to solve a much smaller optimization problem.

1 Introduction

Dimensionality reduction serves to eliminate irrelevant information while preserving the important one. In many cases dimensionality reduction is able to lessen the curse of dimensionality, raise the accuracy rate when there is not enough data (compared to data dimensionality), and improve performance and clustering quality of feature sets. Such improvements are possible since the data lie on or close to a low dimensional manifold, which is embedded in a high dimensional space. Consider, for example, a set of grayscale facial images of resolution $m \times n$ taken under different views with fixed illuminating conditions. Each of the images can be represented with brightness pixel values as a point in $\mathbb{R}^{mn}$ space. However, the intrinsic dimensionality of the manifold formed by these facial images is equal to the degree of freedom of the camera. Therefore, it is much smaller than the image size.

To obtain a relevant low dimensional representation of high dimensional data, several manifold learning algorithms [1, 2, 3, 4, 5] have been recently proposed.

* Olga Kouropteva is grateful to the Infotech Oulu Graduate School and the Nokia Foundation.

H. Kalviainen et al. (Eds.): SCIA 2005, LNCS 3540, pp. 521–530, 2005.

Manifold learning is a perfect tool for data mining that discovers structure of large high dimensional datasets and, hence, provides better understanding of the data. Nevertheless, most of the manifold learning algorithms operate in a batch mode, hence they are unsuitable for sequentially coming data. In other words, when new data arrive, one needs to rerun the entire algorithm with the original data augmented by the new samples.

Recently, an incremental version of one of the manifold learning algorithms called isometric feature mapping (Isomap) [5] has been proposed in [6], where the authors suggested that it can be extended to the online versions of other manifold learning algorithms. Unfortunately, LLE does not belong to this group of algorithms. First of all, as remarked in [7], it is much more challenging to make LLE incremental than other manifold learning algorithms. Secondly, LLE aims at bottom eigenvectors and eigenvalues rather than at top ones. It is well known that ill-conditioning of eigenvalues and eigenvectors frequently occurs in the former case and it is impossible in the latter case. Ill-conditioning means that eigenvalues or/and eigenvectors are susceptible to small changes of a matrix for which they are computed. As a result, problems one faces with when making LLE incremental are more formidable than those for other manifold learning algorithms searching for the top eigenvalues/eigenvectors. This leads to the necessity of inventing another generalization method for LLE. In this paper we propose such a method, called incremental LLE, which is based on the intrinsic properties of LLE. Additionally, we compare the incremental LLE with two previously proposed non-parametric generalization procedures for LLE [8, 9]. Promising and encouraging results are demonstrated in the experimental part.

The paper is organized as follows. A brief description of the LLE algorithm is given in Section 2. Section 3 presents all incremental versions of LLE, including the new one. They are compared on several datasets and the obtained results are discussed in Section 4. Section 5 concludes the paper.

2 Locally Lineal Embedding Algorithms

As input, LLE requires N D dimensional points (one point per pattern) assembled in a matrix $\mathbf{X}$: $\mathbf{X} = \{x_1, x_2, ..., x_N\}, x_i \in \mathbb{R}^D, i = 1, ..., N$. As output, it produces N d dimensional points ($d << D$) assembled in a matrix $\mathbf{Y}$: $\mathbf{Y} = \{y_1, y_2, ..., y_N\}, y_i \in \mathbb{R}^d, i = 1, ..., N$. The i^{th} column of $\mathbf{X}$ corresponds to the i^{th} column of $\mathbf{Y}$.

The LLE algorithm consists of three steps [4]:

1. For each $x_i \in \mathbb{R}^D, i = 1, ..., N$ find its K nearest neighbors: $x_i^1, x_i^2, ..., x_i^K$ by using the Euclidean distance as a similarity measure. A technique for selecting the optimal parameter K for LLE was proposed in [10].
2. Compute weights that best reconstruct each x_i from its K nearest neighbors, $x_i^1, x_i^2, ..., x_i^K$, by minimizing the following cost function:

$$\varepsilon(\mathbf{W}) = \sum_{i=1}^{N} \parallel x_i - \sum_{j=1}^{N} w_{ij} x_j \parallel^2 \ , \tag{1}$$

subject to constraints: $w_{ij} = 0$, if $x_j \notin \{x_i^1, x_i^2, ..., x_i^K\}$, and $\sum_{j=1}^{N} w_{ij} = 1$. The first (sparseness) constraint assures that each point x_i is reconstructed only from its neighbors, while the second condition enforces translation invariance of x_i and its neighbors. Moreover, as follows from Eq. 1, the constrained weights are invariant to rotation and rescaling, but not to local affine transformations, such as shears.

3. Set d - the number of dimensions in the embedding space (see [11] for automatic computing of d). Fix the reconstruction weights w_{ij} and compute low-dimensional embeddings by minimizing the embedding cost function:

$$\delta(\mathbf{Y}) = \sum_{i=1}^{N} \parallel y_i - \sum_{j=1}^{N} w_{ij} y_j \parallel^2, \tag{2}$$

subject to $\frac{1}{N}\sum_{i=1}^{N} y_i y_i^T = I$ (normalized unit covariance) and $\sum_{i=1}^{N} y_i = 0$ (translation-invariant embedding), which provide a unique solution.
Finding a low dimensional embedding under these constraints is equivalent to computing the bottom $d+1$ eigenvectors associated with the $d+1$ smallest eigenvalues of a sparse and symmetric matrix $\mathbf{M} = (I - \mathbf{W})^T(I - \mathbf{W})$. The first eigenvector (composed of 1's) whose eigenvalue is close to zero is excluded. The remaining d eigenvectors yield the final embedding $\mathbf{Y}$.

3 Incremental LLE

LLE operates in a batch or offline mode, that is, it obtains a low-dimensional representation for a certain number of high-dimensional data points to which the algorithm is applied. When new data points arrive, one needs to completely rerun the original LLE for the previously seen dataset augmented by the new data points. In other words, the original LLE lacks generalization to new data. This makes the algorithm to be less attractive especially for large datasets of high dimensionality in a dynamic environment, where a complete rerun of LLE becomes prohibitively expensive.

In [12], an attempt was made to adapt LLE to a situation when the data come incrementally point-by-point. Two simple techniques were proposed, in which the adaptation to a new point can be done either by updating the weight matrix $\mathbf{W}$ or the cost matrix $\mathbf{M}$, respectively. In both these cases, an expensive eigenvector calculation is required for each query. There are two ways to lower the complexity of the LLE generalization: 1) to derive and use a transformation between the original and projected data, and 2) to solve an incremental eigenvalue problem. In this section, we describe two known generalization algorithms belonging to the former case, and propose the incremental version of LLE that uses the latter approach.

Suppose we are given already processed data $\mathbf{X} = \{x_1, x_2, ..., x_N\}$, corresponding projected points $\mathbf{Y} = \{y_1, y_2, ..., y_N\}$, and a new point $x_{N+1} \in \mathbb{R}^D$, which is sampled from the same data manifold as $\mathbf{X}$. We are asked to find a new embedding point $y_{N+1} \in \mathbb{R}^d$ corresponding to the point x_{N+1}.

3.1 Linear Generalization

In order to obtain the new embedded coordinates, the LLE intuition is used, i.e. any nonlinear manifold can be considered as locally linear. This linearity is used to build a linear relation between high and low dimensional points belonging to a particular neighborhood of the data. There are two possibilities of linear generalization [8, 9]:

1. Let us put the K nearest neighbors of x_{N+1} and the corresponding embedded points into the matrices: $\mathbf{X}^{N+1} = \{x^1_{N+1}, x^2_{N+1}, ..., x^K_{N+1}\}$ and $\mathbf{Y}^{N+1} = \{y^1_{N+1}, y^2_{N+1}, ..., y^K_{N+1}\}$. By taking into consideration the assumption that the manifold is locally linear, the following equation is approximately true: $\mathbf{Y}^{N+1} = \mathbf{Z}\mathbf{X}^{N+1}$, where $\mathbf{Z}$ is an unknown linear transformation matrix of size $d \times D$, which can be straightforwardly determined as $\mathbf{Z} = \mathbf{Y}^{N+1}(\mathbf{X}^{N+1})^{-1}$. Because $\mathbf{X}^{N+1}$ is the neighborhood of x_{N+1} and LLE preserves local structures, i.e. points close in the original space remain close in the embedded space, the new projection can be found as $y_{N+1} = \mathbf{Z}x_{N+1}$. Here we multiply the new input by the found transformation matrix, since the underlying manifold must be well sampled and, hence, the neighboring points give sufficient information about the new point [8].
2. To find y_{N+1}, first, the K nearest neighbors of x_{N+1} are detected among points in the high dimensional space: $x_i \in \mathbf{X}, i = 1, ..., N$. Then, the linear weights, w_{N+1}, that best reconstruct x_{N+1} from its neighbors, are computed by using Eq. 1 with the sum-to-one constraint: $\sum_{j=1} w_{N+1j} = 1$. Finally, the new output y_{N+1} is found: $y_{N+1} = \sum_{j=1} w_{N+1j} y_j$, where the sum is over the y_i's corresponding to the K nearest neighbors of x_{N+1} [9].

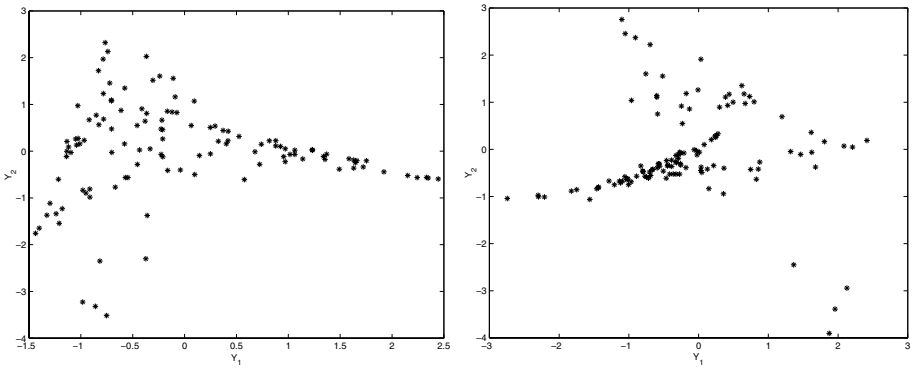
3.2 Incremental LLE

In order to construct embedding LLE searches for the smallest eigenvalues and corresponding eigenvectors of the Hermitian matrix of size $N \times N$. Hence, one has to deal with ill-conditioned eigenproblem [13]. Ill-conditioning means that eigenvalues and/or eigenvectors of a particular matrix are very sensitive to small perturbations of the matrix. For example, changing of the matrix in norm by at most ϵ can change any eigenvalue by at most ϵ, i.e. computing $\lambda_i = 10^{-5}$ to within plus or minus $\epsilon = 10^{-4}$ means that no leading digits of the computed λ_i may be correct.

Eigenvectors and eigenspaces they span are ill-conditioned if small change of the matrix, e.g. changing

$$A_0 = \begin{pmatrix} 2 & 0 & 0 \\ 0 & 1 & \epsilon \\ 0 & \epsilon & 1 \end{pmatrix} \text{ to } A_1 = \begin{pmatrix} 2 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1+\epsilon \end{pmatrix}$$

rotates the two eigenvectors corresponding to the two eigenvalues near one by $\pi/4$, no matter how small ϵ is. Thus they are very sensitive to small changes. Note, that if the eigenvalues are ill-conditioned, then the corresponding eigenvectors are also ill-conditioned; the opposite is not always true.



a) b)

Fig. 1. 2D spaces formed by two bottom eigenvectors of a) the initial matrix, and b) the modified matrix. The difference between norms of these matrices is equal to $-2.36 \cdot 10^{-33}$

Fig. 1 demonstrates the consequence of ill-conditioning. First, we find two bottom eigenvectors of a particular matrix. These eigenvectors form 2D space in Fig. 1 (a). Then we change the elements of the matrix by adding or subtracting very small values. Finally, we compute two bottom eigenvectors of the modified matrix, which are shown in Fig. 1 (b). One can see that these plots dramatically differ from each other, while the difference between norms of the initial and modified matrices is equal to $-2.36 \cdot 10^{-33}$.

When a new data point arrives, the main goal of the incremental LLE is to compute the new cost matrix $\mathbf{M}_{new}$ to be exactly the same as if it would be computed by LLE applied to the old data augmented by the new data point. This can be done by applying the following operations: first, distances between points, which either belong to the K nearest neighbors of the new point or contain the new point as one of their K nearest neighbors, are recalculated. Then the weights for the points whose distances have been changed are updated by solving Eq. 1 and the new matrix $\mathbf{M}_{new}$ of size $(N+1) \times (N+1)$ is calculated by using these weights.

The classical eigenproblem is defined as the solution of the equation $\mathbf{M}y_i^T = \lambda_i y_i^T$ or in matrix form $\mathbf{M}\mathbf{Y}^T = diag\{\lambda_1, \lambda_2, ..., \lambda_d\}\mathbf{Y}^T$. Since typical eigenvectors are orthogonal, we can rewrite the eigenproblem: $\mathbf{Y}\mathbf{M}\mathbf{Y}^T = diag\{\lambda_1, \lambda_2, ..., \lambda_d\}$. Without loss of generality, we assume that the eigenvalues of the new cost matrix, $\mathbf{M}_{new}$ are the same as for the cost matrix computed for N points. This can be done since the eigenvalues, we are dealing with, are very close to zero, usually they are of order 10^{-p}, where p is large enough (practically about 10). Therefore we can write $\mathbf{Y}_{new}\mathbf{M}_{new}\mathbf{Y}_{new}^T = diag\{\lambda_1, \lambda_2, ..., \lambda_d\}$, where $\{\lambda_i\}, i = 1, ..., d$ are the smallest eigenvalues of the cost matrix computed for N points. The new coordinates are obtained by solving $d \times d$ minimization problem:

$$\min_{\mathbf{Y}_{new}} (\mathbf{Y}_{new}\mathbf{M}_{new}\mathbf{Y}_{new}^T - diag\{\lambda_1, \lambda_2, ..., \lambda_d\}). \quad (3)$$

The LLE constraints imposed on the embedding coordinates should be kept. Thus, the $N \times N$ problem of the third LLE step was reduced to the $d \times d$ problem, where $d << N$. Since d is usually very small, say 10 or so, the minimization is not time consuming and can be done for every arriving point.

4 Experiments

In the experiments we applied the three LLE generalization algorithms described in the previous section to the datasets represented in Table 1. The Olga's and Oleg's faces datasets were taken by a Sony DFW-X700 digital camera under fixed illumination conditions. Each sequence consists of images showing one person slowly rotating the head from left to right, while trying to fix a chin at the same level in order to obtain one degree of freedom: angle of rotation. In spite of the fact that initial conditions for capturing these datasets were the same, the Oleg's faces data is uniformly distributed, while Olga's faces data is not. This is due to the velocity of head rotation: Olga rotated her head from the left to frontal view slower that from frontal view to the right; therefore, there are more frames for the former case than for the latter one. Hence, we consider Olga's faces dataset to be non-uniform. The description of other datasets can be found from the corresponding references.

Table 1. Datasets used in the experiments

Data	N points	Dimensionality	Features
Swissroll [9]	2000	3	Coordinates
S-curve [9]	2000	3	Coordinates
Wine [14]	178	13	Chemical measurements
Fray faces [15]	1965	560	Grayscale pixels values
MNIST digits(3&8) [16]	1984	784	Grayscale pixels values
Coil-20 [17]	1440	4096	Grayscale pixels values
Oleg's faces	1130	6300	Grayscale pixels values
Olga's faces	1200	6300	Grayscale pixels values

All datasets were divided into training (70%) and test (30%) sets. The training sets were projected by LLE to two dimensional spaces and the test sets were mapped to the corresponding space by the generalization algorithms described in Section 3.

The first experiment is done with the swissroll dataset (Fig. 2 (a)). At the beginning, we project the initial data containing 1,400 points by the conventional LLE algorithm ($K = 15$), and then we add the test data points in a random order. In Fig. 2 (b, c, d) results of generalizing ten new data points are shown. As one can see the projections of the new points are visually almost the same for all methods.

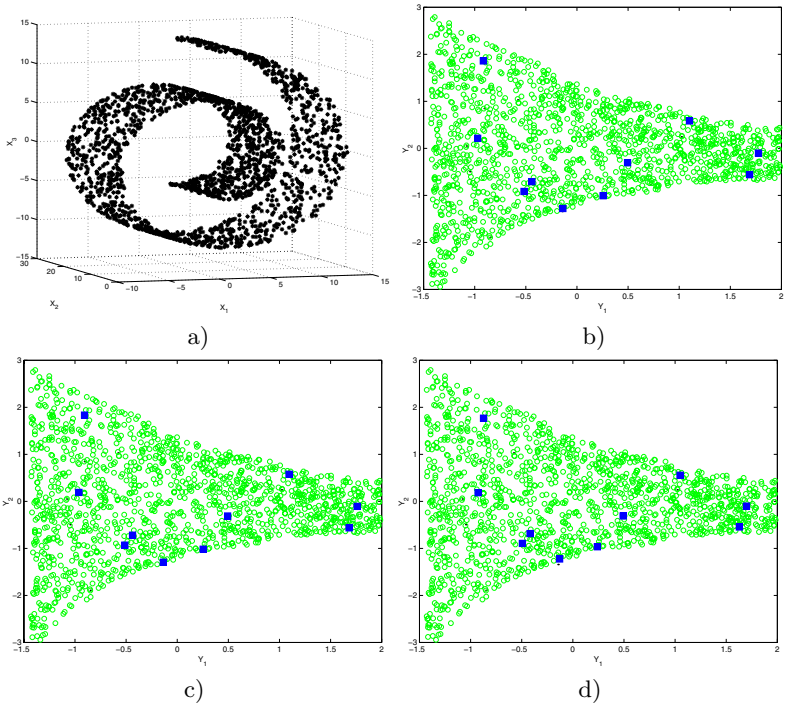
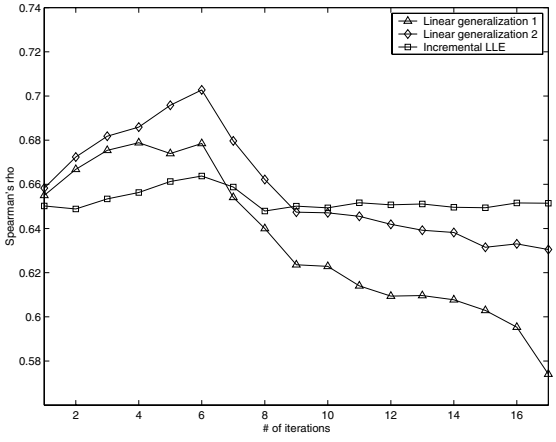


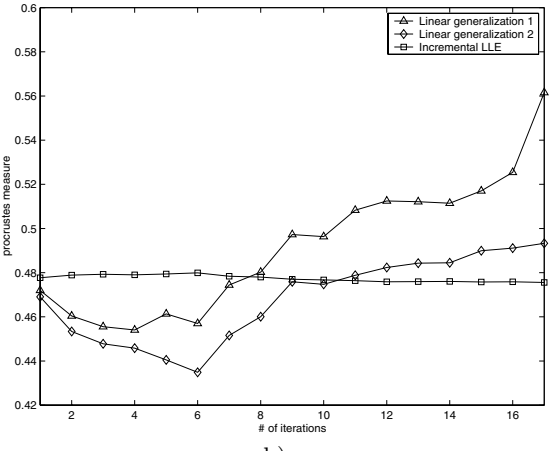
Fig. 2. LLE generalization on the swissroll dataset. (a) The original 3D data points sampled from the corresponding data manifold. The circles (∘) depict the target coordinates, i.e. those, which are obtained by applying the conventional LLE to the pooled dataset, including both old and new data points. The dots (·) show the estimated coordinates for b) linear generalization 1; c) linear generalization 2; d) incremental LLE. The filled squares (□) correspond to the projections of the new points

That is why, in order to quantitatively compare the generalization methods, we calculate two characteristics, namely, Spearman's rho and procrustes measure. The Spearman's rho estimates the correlation of rank order data, i.e. how well the corresponding low dimensional projection preserves the order of the pairwise distances between the high dimensional data points converted to ranks. The best value of Spearman's rho is equal to one. In its turn, the procrustes measure determines how well a linear transformation (translation, reflection, orthogonal rotation, and scaling) of the points in the projected space conforms to the points in the corresponding high dimensional space. The smaller the value of procrustes measure, the better fitting is obtained. Both Spearman's rho and procrustes measure are commonly used for estimating topology preservation when doing dimensionality reduction.

In another experiment, 119 wine data training points are projected by the conventional LLE ($K = 15$) and the other 51 test points are added during 17 iterations ($n = 3$ points per time) by three generalization methods: linear



a)



b)

Fig. 3. Spearman's rho (a) and procrustes measure (b) for the wine dataset

generalization 1 (LG1), linear generalization 2 (LG2), and incremental LLE (ILLE). In Fig.3 plots show the Spearman's rho and procrustes measure estimated after each iteration. One can see that the incremental LLE performs better than other generalization procedures when the number of new samples increases. But this is not always the case. In order to make the final conclusion, we estimate the Spearman's rho and procrustes measure after each iteration for all datasets and count the number of iterations for which a particular method outperforms others in terms of the Spearman's rho and procrustes measure. The number of iterations, number of points per iteration, and the results for all datasets are listed in Table 2. The largest resulting values are underlined.

By looking at Table 2, a number of interesting observations can be done.

Table 2. Spearman's rho (ρ_{Sp}) and procrustes measure ($Procr$) for the datasets

Data	N of iterations	N of points per iteration		LG1	LG2	ILLE
S-curve	60	10	ρ_{Sp}	47	12	1
			$Procr$	43	16	1
Wine	17	3	ρ_{Sp}	0	8	9
			$Procr$	0	10	7
Fray faces	28	21	ρ_{Sp}	1	8	19
			$Procr$	0	14	14
MNIST digits(3&8)	22	27	ρ_{Sp}	0	0	22
			$Procr$	0	22	0
Coil-20	27	16	ρ_{Sp}	0	27	0
			$Procr$	1	5	21
Oleg's faces	33	10	ρ_{Sp}	2	31	0
			$Procr$	16	17	0
Olga's faces	36	10	ρ_{Sp}	0	6	30
			$Procr$	14	5	17

- When manifold is well and evenly sampled, and the relationships between original and embedded datasets are close to be locally linear as in case of S-curve and Oleg's faces, LG1 and especially LG2 are sufficient for successful generalization of test points, and this fact is confirmed by both Spearman's rho and procrustes measure.
- However, if the data manifold is non-uniformly sampled and the relationships are locally nonlinear as in case of Olga's and Fray's faces, ILLE emerges as a clear winner, since it does not rely on linear relationships. This is again supported by both Spearman's rho and procrustes measure.

5 Conclusion

LLE belongs to the class of manifold learning algorithms, which reduce dimensionality by learning structure of data manifolds. The deficiency of LLE is that if new inputs arrive, one needs to rerun the algorithm for the pool of the old and new data. The main difficulty of making LLE incremental is that it obtains embeddings by searching for the smallest eigenvectors, which are ill-conditioned. In this paper we proposed a new method for LLE generalization, called incremental LLE, and compared it with linear generalization methods. The results demonstrate that ILLE is a powerful generalization method for data whose distribution is not uniform and the local linearity constraints do not hold. In contrast, the linear generalization methods deal perfectly with well-sampled manifolds of artificially generated data but may have problems with real-world datasets.

One of the directions of the future work is to compare the classification performance of the linear generalization algorithms and ILLE on different datasets.

References

1. Belkin, M., Niyogi, P.: Laplacian eigenmaps for dimensionality reduction and data representation. Technical Report TR-2002-01, University of Chicago, Department of Computer Science (2002)
2. DeCoste, D.: Visualizing Mercer kernel feature spaces via kernelized locally-linear embeddings. In: Proc. of the 8th Int. Conf. on Neural Information Processing, Shanghai, China. (2001)
3. Donoho, D., Grimes, G.: Hessian eigenmaps: new locally linear embedding techniques for high-dimensional data. Proc. of National Academy of Sciences **100** (2003) 5591–5596
4. Roweis, S., Saul, L.: Nonlinear dimensionality reduction by locally linear embedding. Science **290** (2000) 2323–2326
5. Tenenbaum, J., de Silva, V., Langford, J.: A global geometric framework for nonlinear dimensionality reduction. Science **290** (2000) 2319–2323
6. Law, M., Zhang, N., Jain, A.: Nonlinear manifold learning for data stream. In Berry, M., Dayal, U., Kamath, C., Skillicorn, D., eds.: Proc. of the 4th SIAM International Conference on Data Mining, Lake Buena Vista, Florida, USA. (2004) 33–44
7. Bengio, Y., Paiement, J.F., Vincent, P., Delalleau, O., Le Roux, N., Ouimet, M.: Out-of-sample extensions for lle, isomap, mds, eigenmaps, and spectral clustering. In Thrun, S., Saul, L., Schölkopf, B., eds.: Advances in Neural Information Processing Systems 16, Cambridge, MA, MIT Press (2004)
8. Kouropteva, O., Okun, O., Hadid, A., Soriano, M., Marcos, S., Pietikäinen, M.: Beyond locally linear embedding algorithm. Technical Report MVG-01-2002, University of Oulu (2002)
9. Saul, L., Roweis, S.: Think globally, fit locally: unsupervised learning of nonlinear manifolds. Journal of Machine Learning Research **4** (2003) 119–155
10. Kouropteva, O., Okun, O., Pietikäinen, M.: Selection of the optimal parameter value for the locally linear embedding algorithm. In: Proc. of 2002 Int. Conf. on Fuzzy Systems and Knowledge Discovery, Singapore. (2002) 359–363
11. de Ridder, D., Duin, R.: Locally linear embedding for classification. Technical Report PH-2002-01, Delft University of Technology (2002)
12. Kouropteva, O.: Unsupervised learning with locally linear embedding algorithm: an experimental study. Master's thesis, University of Joensuu, Finland (2001)
13. Bai, Z., Demmel, J., Dongorra, J., Ruhe, A., van der Vorst, H.: Templates for the solution of algebraic eigenvalue problems. SIAM, Philadelphia (2000)
14. (http://www.ics.uci.edu/∼mlearn/)
15. (http://www.cs.toronto.edu/∼roweis/data.html)
16. (http://yann.lecun.com/exdb/mnist/index.html)
17. (http://www1.cs.columbia.edu/CAVE/research/softlib/coil 20.html)